An Adaptive Cartesian Grid Projection Method for Environmental Flows

by

Michael Frederick Barad
B.S. (University of Colorado at Boulder) 1993
M.S. (University of California, Berkeley) 1997

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Civil and Environmental Engineering

in the

OFFICE OF GRADUATE STUDIES
of the
UNIVERSITY OF CALIFORNIA
DAVIS

Approved:

_____

_____

_____

Committee in charge

2006

An Adaptive Cartesian Grid Projection Method for Environmental Flows

by

Michael Frederick Barad

Abstract

An Adaptive Cartesian Grid Projection Method for Environmental Flows

by

Michael Frederick Barad

Doctor of Philosophy in Civil and Environmental Engineering

University of California, Davis

Professor S. Geoffrey Schladow, Chair

In this work we present our block-structured adaptive mesh refinement (AMR) computational fluid dynamics model. The model is based on the solution of the unsteady, incompressible, Navier-Stokes equations in two or three dimensions, including air/water and fluid/solid interfaces and the transport of scalars. The methodology is based on a second-order accurate projection method with high-order accurate Godunov finite differencing including slope limiting and a stable differencing of the nonlinear convection terms. This is a proven methodology for hyperbolic problems that yields accurate transport with low phase error while minimizing the numerical diffusion at steep gradients typically found in classical high order finite difference methods. This methodology is combined with finite volume AMR discretizations based on flux matching at refinement boundaries to obtain a conservative method that is second-order accurate in solution error for environmental flows. The control volumes are formed by the intersection of the irregular embedded boundary with Cartesian

grid cells. Unlike typical discretization methods, these control volumes naturally fit within easily parallelized disjoint block data structures, and permit dynamic AMR coarsening and refinement as a simulation progresses. AMR allows the simulation of a range of spatial and temporal scales. Capturing these ranges is critical to accurately modeling multiscale transport complexities such as boundaries, fronts, and mixing zones that exist in natural environments.

To Katie

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank my advisor Professor Geoffrey Schladow for his guidance and support during this research. Thanks to Dr. Phillip Colella for training me as a computational scientist and for providing me with a spectacular working environment within his Applied Numerical Algorithms Group (ANAG) at Lawrence Berkeley National Laboratory (LBNL). I thank Professor Elbridge Gerry Puckett for introducing me to the mathematical methods of this work.

Thanks to all of the researchers in the ANAG at LBNL for their support, wisdom and friendship. This work would not have been achievable without their help, and is truly a team effort. Specifically, I thank Dr. Daniel Graves for helping me out of more tough spots than I wish to reveal, Dr. Daniel Martin for his computational wisdom, Dr. Terry Ligocki for patiently helping me get around tricky situations, Dr. Peter Schwartz for always wanting to help me solve yet another math problem, Dr. David Serafini for being my emacs and build system goto wizard, George "Chip" Smith for keeping our computers running smoothly, Dr. Theodore Sternberg for continuously improving our visualization software, and Brian Van Straalen for his behind the scenes hard work.

I thank Professor Oliver Fringer for introducing me to internal wave problems, and for his contribution to Section 6.5.1. Thanks to Dr. David Schoellhamer and his USGS field crew for aiding with our San Pablo Bay experiments. I thank Dr. David Trebotich for his help in developing some of the algorithms. Thanks to Dr. Bassam Younis for many interesting discussions.

I want to thank past and present members of the Environmental Dynamics Labo-

ratory for aiding me in my research and for their friendship. Specifically, I thank Dr. Sveinn Palmarsson, Dr. Francisco Rueda, Todd Steissberg for his help with Lake Tahoe remote sensing and bathymetry data, and Dr. John Warner for his help in the San Pablo Bay field experiments.

I thank my best friend Will Forney for encouraging me to pursue this dream with a few fun distractions along the way. I thank my first academic mentor Dr. Pedro Restrepo for introducing me to computational hydrology. I thank Dr. Jeffrey Haltiner for stoking my hydrologic interests even further.

Thanks to all my family for their fantastic support throughout this long journey. Thanks to my mother Vera for always believing in me. Thanks to my late step-father Ed Marks who encouraged me to do my best. Thanks to my sisters, Amelia and Vivi, and their blooming families: Todd, Desmond, Wyatt, Jason, and our newest edition Esther. Thanks to my fiancee Katie, whose love has inspired me and with whom I look forward to sharing an adventurous life.

Funding for this research was provided by a Department of Energy (DOE) Computational Science Graduate Fellowship (CSGF), a UCTSR&TP fellowship, and by the Environmental Protection Agency through the CISNet project. Computer time was gratefully provided by the DOE at both LBNL and the National Energy Research Scientific Computing Center.

Chapter 1

Introduction

1.1 Motivation

We want to develop an analysis capability to study highly nonlinear, multiscale flows in oceans, lakes, and rivers that are well represented by the incompressible Navier-Stokes equations. Examples of such flows are: internal waves, coastal plumes, density currents in lakes, flows in branched estuarine slough networks, and flows past highly complex topography. Issues involved include complex and often sparse geometries, large ranges in spatial and temporal scales, moving fronts and highly complex mixing zones. We hope to provide a predictive tool for both engineering and science with an enhanced ability to interpret and extend the results of field and laboratory studies.

1.2 Review of Previous Work

1.2.1 Models

Historically, computational environmental fluid mechanics methods have focused on simplifying either the governing equations or the dimensionality of the problem (and often both). The equations have been reduced to as simple as the one dimensional Bernoulli equations, yet most numerical methods solve either shallow-water, hydrostatic, and/or Boussinesq approximations in one, two, or three dimensions.

The main benefit of shallow-water models is that they are computationally very efficient, this is due to the hyperbolic nature of the shallow-water equations. The major drawback of shallow-water models is that they crudely approximate complex flows in one dimension (1D) or two dimensions (2D), where three-dimensional (3D) effects are entirely neglected.

Hydrostatic models are able to represent fluid motion in 3D, yet the vertical momentum balance is replaced by a hydrostatic balance between vertical pressure gradients and buoyancy forcing. One of the main benefits of 3D hydrostatic models is that they are computationally more efficient than nonhydrostatic models because the elliptic problems are two-dimensional. Unfortunately, the hydrostatic approximation entirely neglects vertical accelerations in the vertical momentum balance; this is highly problematic for studying complex, highly non-linear 3D motions [70, 105].

The Boussinesq approximation is very common in numerical models of geophysical hydrodynamics. The Boussinesq approximation assumes that density variations are small compared to a reference fluid density. Boussinesq models neglect density variations in all

terms but the buoyancy forcing. For many problems, the Boussinesq approximation works well for variations in density that are less than roughly four percent [66]. For highly non-linear, density driven flows with density variations more than four percent, the Boussinesq approximation can introduce subtle errors that deviate from a full incompressible Navier-Stokes solution [66, 49].

Our approach will solve the complete incompressible Navier-Stokes equations in either 2D or 3D, with a full variable density forcing (except on the viscous term where we invoke a Boussinesq approximation). For a more detailed summary of 3D geophysical hydrodynamic models see [95].

1.2.2 Temporal Discretizations

There is a range of approaches for solving the incompressible Navier-Stokes equations. Methods include artifical compressibility methods (see [29]), Lagrangian vortex methods (see [28]), Galerkin finite element methods (see [50, 20]), and different flavors of projection methods.

Projection methods are a powerful solution technique for the incompressible Navier-Stokes equations pioneered in a series of papers by Chorin [25, 26, 27]. In Chorin's work he developed a numerical projection method using a discrete form of the Hodge decomposition. In projection methods the solution is advanced by predicting a velocity field that does not satisfy a discrete divergence constraint, and then correcting the predicted velocity so that it is approximately divergence-free. In the predictor step, it is necessary to explicitly approximate the advective terms, and (for viscous problems) to subsequently solve a parabolic equation implicitly for the predictor velocity. The correction is achieved by solving an el-

liptic equation implicitly via a Hodge decomposition [30]. For the parabolic solves in the predictor step, researchers have used first-order accurate backward-Euler methods, second-order accurate Crank-Nicolson [9], and other implicit Runge-Kutta methods [104, 75].

Chorin's original work was based on first-order accurate in time discretizations. The projection method was then extended to second-order accuracy in time by Bell et. al. [9]. The method was further extended to variable density flows in [12], and subsequently with adaptive mesh refinement in [1, 72]. Various other researchers have extended the projection method, including [48, 102, 46, 78, 64, 87, 99] among others. In this research we will extend the approach of [9] to complex geometry with variable density and adaptive mesh refinement. See [21] for a more detailed history of projection methods.

1.2.3 Spatial Discretizations

Discretization Methods

To study fluid systems, researchers typically use one of two reference frames: Eulerian or Lagrangian. The Lagrangian reference frame follows individual fluid parcels as they move through space and time, while the Eulerian reference frame focuses on fixed regions in space where fluid may pass. For this work we use the Eulerian reference frame. A discussion of Lagrangian methods is beyond the scope of this work, the interested reader will find [79, 37, 68, 55] informative. With an Eulerian method, researchers typically discretize space using one of three methods: finite element, finite difference, or finite volume.

Typical finite element methods have the benefit of irregular elements. These elements have the attractive characteristic that, with refinement they can accurately fit arbi-

trarily complex geometries, yet they necessitate unstructured data structures for the entire flow field (which are discussed below), and typically lead to non-conservative discretizations.

The most basic finite difference methods utilize "stair-step" (i.e. in or out rectangular volumes) approximations to irregular geometry and suffer from a catastrophic lack of accuracy for non-orthogonal geometries (see Appendix A.3). Immersed boundary and ghost-fluid methods [86, 51, 47] are an attractive finite difference method (due to their implementation simplicity) but suffer due to conservation issues stemming from a non-flux based approach.

The finite volume method is the most attractive of the spatial discretization methods due to its conservation properties. In finite volume methods the flow domain is decomposed into individual control volumes.

Mapped grid based control volumes are a very promising method yet they are not able to handle arbitrarily complex geometries (for lack of a mapping function). The Embedded Boundary (or cut-cell) method [60, 75, 92, 31] is an elegant technique where control volumes are formed by the intersection of the domain with rectangular grid cells. The intersections can be computed to arbitrary accuracy, and are generated with an $O(N^{\frac{D-1}{D}})$ algorithm from distance type functions (e.g. a Digital Elevation Model). For the bulk of the flow domain we use a structured grid, for the irregular control volumes (on a codimension one less than the bulk) we use an unstructured mapping within each (domain decomposition) box. With this embedded boundary spatial description, the geometry can be arbitrarily complex, yielding tractable, efficient and accurate conservative finite volume discretizations. In this research we use a finite volume method where our control volumes

are of the Embedded Boundary (EB) type.

Structured and Unstructured Gridding

If the control volumes are computationally represented on a rectangular array, or on a union of rectangular arrays the mesh is termed structured, otherwise the mesh is termed unstructured. Meshes that can change as a simulation progresses are referred to as temporally adaptive. Meshes that are not uniform throughout the spatial domain are referred to as spatially adaptive.

Models that use unstructured meshes have cumbersome data structures that are computationally less efficient to access and require more complex mathematical techniques to maintain accuracy. While unstructured mesh models have the benefit of spatial adaptivity, temporally adaptive fully unstructured numerical methods are still an open question. Non-adaptive, structured mesh codes utilize highly efficient data structures but require global refinement which is prohibitively costly for multiscale flows. The most successful structured, spatially and temporally adaptive codes use blocks to partition the domain into regions of equal refinement. Block structured adaptive codes benefit from highly efficient data structures, the ability to locally refine, and well understood conservative finite volume discretization methods. Excluding this work, all known existing computational environmental fluid mechanics models use meshes that are non-adaptive in time.

1.3 Proposed Method

In this section the key features of our method for solving the incompressible Navier-Stokes equations are outlined. Specifically, the governing equations, accuracy of the method, spatial discretization, and implementation are discussed.

1.3.1 Second-Order Accurate Projection Method For Incompressible Navier-Stokes

Instead of shallow-water, hydrostatic, or Boussinesq approximations that are typically found in geophysical hydrodynamic models, this research solves the incompressible variable-density Navier-Stokes equations, following [12], and exhibits second-order accurate convergence in time and space. Note that we treat our viscous term with a Boussinesq approximation, but, due to the nearly inviscid nature of geophysical flows, this is an insignificant approximation. Solving the nonhydrostatic equations is essential for accurately simulating flows with pressure distributions that significantly deviate from hydrostatic (i.e. $p \neq \rho g z + p_o$). Avoiding the (inviscid) Boussinesq approximation permits the simulation of geophysical flows where density deviations are significant enough to alter the motions through more than just the buoyancy term.

The algorithms in this research yield second-order accurate solution errors for the governing equations. For second-order methods, when the mesh spacing is refined by a factor of two, the error is reduced by a factor of four. Higher-order methods can resolve flow features that lower-order methods can only resolve with significantly finer (and therefor less efficient) grids. A common claim in the geophysical hydrodynamic modeling

community is that numerical methods are r-order accurate (r=1,2) without backing this up with convergence studies for realistic problems. In this work we systematically show second-order accurate results for both the de-coupled operators and for a range of incompressible Navier-Stokes test problems.

1.3.2 Accurate Spatial Description with Embedded Boundaries

In this research the irregular domain is discretized as a collection of control volumes formed by the intersection of the domain with rectangular grid cells. With this efficient and accurate embedded boundary spatial description, the geometry can be arbitrarily complex. This geometric capability permits the accurate study of a variety of complex fluid flows such as those that occur on coastal shelves, in branched estuarine slough networks, in complex closed conduits, and even past the detailed structures of benthic invertebrates. The current generation of numerical methods for modeling environmental flows is severely limited in their ability to model such complex geometries.

1.3.3 Multiscale Capturing with Adaptive Mesh Refinement

Adaptive methods for the numerical solution of partial differential equations concentrate computational effort when and where it is most needed. For over two decades block-structured adaptive mesh refinement (AMR) has proven useful for overcoming limitations in computational resources and accuracy in problems outside of the environmental fluid mechanics community in such fluid fields as astrophysics [10, 35], combustion [84, 11], atmospheric science [94], applied mathematics [1, 5, 13, 8, 14, 16, 100] aerospace engineering [15] and ink jet printer design [99]. In environmental fluid mechanics, most prior work

has focused on static, one-way nested refinement strategies, where finer grids are nested within coarser grids and these grids do not adapt in time. In one-way nesting, there is no feedback between the fine and coarse grids [85], while in two-way nesting, the solution on the fine grid is coupled to the solution on the coarse grid [106, 42, 1, 72]. Conservative, flux based two-way AMR coupling has desirable well posedness properties, e.g. divergence operators have desirable telescoping sum properties (see [5]).

AMR with conservative two-way nesting provides the capability to simultaneously capture scales ranging many orders of magnitude (e.g. from hundreds of kilometers to meters). Consider oceanic internal waves, AMR works by placing coarse grids over the entire flow domain, and recursively finer nested grids adaptively track the generation, propagation, interaction and dissipation of internal waves. With AMR, one can "zoom in" on moving regions and accurately capture the important flow physics at multiple scales. Accurately capturing a range of spatial and temporal scales is critical to accurately simulating complex environmental flows.

The embedded boundary approach naturally fits within easily parallelized disjoint block data structures, and permits dynamic AMR coarsening and refinement as a simulation progresses (see Figure 3.9 for an example embedded boundary AMR grid). The physical processes that we wish to study contain spatial scales that span meters to hundreds of kilometers, with temporal scales ranging from seconds to days. The only way to accurately resolve this range of scales is to use parallel adaptive mesh refinement on high performance computers. With AMR the mathematical model tracks important flow features with finer grids as they evolve throughout the spatial and temporal flow domain, expending compu-

tational resources only where and when they are needed. The application of AMR to the study of environmental flows is a new and powerful technique. In this research we use fully adaptive, block structured, two-way nested meshes.

1.3.4 Implementation

This work was implemented within, and extended the Chombo [32] adaptive mesh refinement framework. Chombo provides the necessary data structures to implement the highly complex adaptive algorithms required for solving the incompressible Navier-Stokes equations in irregular domains. Chombo builds and executes on a range of computational platforms, from laptops to parallel supercomputers. Chombo is implemented primarily in the C++ programming language. For performance reasons, Chombo also provides an interface to FORTRAN for fast, regular array operations. Chombo also provides an elegant dimension independent programming paradigm that accelerates the development cycle due to both the ease of debugging in reduced dimensions, and due to a single code base for multiple dimensions. This dissertation is a collaborative effort with all the Chombo developers.

1.4 Summary of Contents

In this thesis an adaptive Cartesian grid projection method is presented, and results for idealized and field-scale geophysical flows are presented.

Chapter 2 describes the governing equations that we solve and the basic mathematical theory behind the projection method. Chapter 2 also includes boundary conditions, and notations that are used throughout this work. In Chapter 3 we describe our embed-

ded boundary discretization, including our geometric description, and our single-grid finite volume discretizations for elliptic, parabolic, and hyperbolic PDEs. Chapter 4 describes our solution methodology for solving the incompressible Navier-Stokes equations. Chapter 5 describes our extension of the single-grid algorithm to block-structured adaptive mesh refinement. In Chapter 6, we present results from both lab-scale classical fluid mechanics problems and from field-scale geophysical flows. In Chapter 7, we summarize the work and highlight how our results are significant.

This work contains four appendices. Appendix A is a catch all for discretizations that lack a home in Chapter 4. Appendix B presents convergence results for the de-coupled operators. Appendix C presents example grid generation and flow visualizations. Appendix D presents journal papers co-authored during the completion of this dissertation.

Chapter 2

Governing Equations

2.1 Incompressible Navier-Stokes Equations

- Momentum balance,

$$\vec{u}_t + (\vec{u} \cdot \nabla)\vec{u} = -\frac{\nabla p}{\rho} + \vec{g} + \nu \Delta \vec{u} \tag{2.1}$$

- Divergence-free constraint,

$$\nabla \cdot \vec{u} = 0 \tag{2.2}$$

- Density conservation,

$$\rho_t + \vec{u} \cdot \nabla \rho = 0 \tag{2.3}$$

- Passive scalar transport,

$$s_t + \vec{u} \cdot \nabla s = k_s \Delta s + H_s \tag{2.4}$$

We have left out the Coriolis term in the momentum balance. For problems where the Coriolis term is relevant, this term can easily be added to our method by following [48].

2.2 Boundary Conditions

We define our flow domain as Ω, and the boundary of this domain as $\partial\Omega$. We decompose $\partial\Omega$ into n disjoint domain boundaries (D_i), and m disjoint embedded boundaries (E_j). We denote the boundary set as, $\partial\Omega = (\bigcup_{i=1}^{n} D_i) \bigcup (\bigcup_{j=1}^{m} E_j)$. Domain boundaries are only permitted to exist on a specified rectangular parallelepiped Γ, where Ω is contained within Γ. Embedded boundaries are permitted to exist on or within Γ. For each D_i and E_j, we permit different boundary conditions for each variable ($\vec{u}$, p, ρ, and s). We do not permit periodic boundary conditions on E_j.

We define $\vec{n}$ as the unit normal to the boundary which points into the fluid. With $\vec{\tau}_1$, and $\vec{\tau}_2$ unit vectors tangential to the boundary, where $\vec{n} \perp \vec{\tau}_1 \perp \vec{\tau}_2$. The following Sub-Sections describe the boundary conditions in detail.

2.2.1 Velocity Boundary Conditions

On solid surfaces the following velocity boundary conditions are permitted:

1. Dirichlet for any solid surface (EB or domain boundary): $\vec{u} = \vec{b}$
2. Free-slip for domain walls: $\vec{u} \cdot \vec{n} = 0$, $\nabla(\vec{u} \cdot \vec{\tau}_1) \cdot \vec{n} = 0$, and $\nabla(\vec{u} \cdot \vec{\tau}_2) \cdot \vec{n} = 0$.

At inflow boundaries we specify $\vec{u}$. At outflow boundaries and free-surface boundaries we specify $\vec{u} \cdot \vec{n}$ by setting $\nabla \cdot \vec{u} = 0$, unless, for outflow in the hyperbolic operators, this leads to inflow, in which case we set the velocity normal to the domain boundary to zero. For outflow and free-surface in the hyperbolic operators we specify the tangential velocity components by extrapolating normal to the domain face. For outflow in the viscous operators we specify

a zero gradient condition on velocity. The method also permits periodic domain boundary conditions.

2.2.2 Pressure Boundary Conditions

Pressure boundary conditions we prescribe to be compatible with velocity boundary conditions and the governing equations. Specifically, for solid surfaces (where $\vec{u} \cdot \vec{n} = 0$) we prescribe,

$$\frac{\nabla p}{\rho} \cdot \vec{n} = \vec{g} \cdot \vec{n}. \tag{2.5}$$

At inflow faces we prescribe boundary conditions as in (2.5). With a hydrostatic free-surface Dirichlet boundary condition for pressure at $z = 0$,

$$p = p_{atm} - \rho_o g \eta. \tag{2.6}$$

where g is gravitational acceleration. For outflows at boundaries we prescribe hydrostatic pressure,

$$p = p_{atm} - \rho_o g(\eta - z), \tag{2.7}$$

where $z = 0$ is the top plane of Γ (see Γ in Section 2.2) in direction D. The method also permits periodic domain boundary conditions.

Free-Surface Pressure Boundary Condition: Using Shallow-Water Equations

To satisfy (2.6) and (2.7), we need to specify η, the free-surface height (as in [62, 39]). To do this, we first define a depth averaging operator

$$<> \equiv \frac{1}{\eta + H} \int_{-H}^{\eta} dz, \tag{2.8}$$

where we define the unperturbed depth H as the vertical distance from $z = 0$ to the bathymetry. We also define a horizontal divergence

$$\nabla^{\perp} \cdot \vec{F} \equiv \sum_{d' \neq D} \frac{\partial \vec{F}_{d'}}{\partial x_{d'}}, \tag{2.9}$$

a horizontal gradient $\nabla^{\perp}$ which contains all but the vertical component, and a horizontal vector $\vec{F}^{\perp}$ which contains all but the vertical component. If we depth average our divergence-free constraint and apply boundary conditions we obtain,

$$\eta_t + \nabla^{\perp} \cdot ((\eta + H) < \vec{u} >) = 0. \tag{2.10}$$

If we depth average our momentum balance (2.1) we obtain,

$$< \vec{u}_t + (\vec{u} \cdot \nabla)\vec{u} > = < -\frac{\nabla p}{\rho} + \vec{g} + \nu \Delta \vec{u} > \tag{2.11}$$

where if we decompose the pressure: $p \equiv p_{Atmospheric} + p_{Hydrostatic} + p_{Nonhydrostatic}$, this yields,

$$< \vec{u} >_t + < (\vec{u} \cdot \nabla)\vec{u} > = - < \frac{\nabla p_H}{\rho} > - < \frac{\nabla p_N}{\rho} > + \vec{g} + < \nu \Delta \vec{u} > . \tag{2.12}$$

If we neglect all but the hydrostatic and temporal acceleration terms, assume constant density, and only keep the horizontal balance, we have

$$< \vec{u}^{\perp} >_t + < \frac{\nabla^{\perp} p_H}{\rho_o} > \approx 0. \tag{2.13}$$

We recall the hydrostatic pressure is $p_H = -\rho g(\eta - z)$, where g is the gravitational acceleration. This results in a horizontal equation, where the depth averaged velocity is driven solely by free-surface gradients,

$$< \vec{u}^{\perp} >_t - g \nabla^{\perp} \eta \approx 0. \tag{2.14}$$

We evolve the free-surface height η by solving (2.10) and (2.14). We use η to set our free-surface pressure boundary condition (2.6) and (2.7).

Note: all results presented in this work are "rigid-lid" ($\eta = 0$), a description of our un-tested free-surface algorithm is given in Appendix A.4-A.5.

2.2.3 Scalar Boundary Conditions

Scalar boundary conditions we prescribe to be compatible with velocity boundary conditions. Note that density boundary conditions are treated the same as scalars. Specifically, for solid surfaces we prescribe no flux,

$$\nabla s \cdot \vec{n} = 0. \tag{2.15}$$

For inflow boundaries we specify scalars to be some known value,

$$s = f(\vec{x}, t). \tag{2.16}$$

For outflow boundaries in the hyperbolic operator we extrapolate the scalar normal to the boundary. For the free-surface we prescribe no flux through the free-surface. The method also permits periodic domain boundary conditions.

2.3 Projection Formulation

Following the work of Chorin [27, 26, 28, 25, 30] we shall use the Helmholtz-Hodge Decomposition Theorem: A vector field $\vec{w}$ on a region of space Ω can be uniquely decomposed in the form

$$\vec{w} = \vec{u} + \nabla\phi, \tag{2.17}$$

$$\Delta\phi = \nabla \cdot \vec{w} \tag{2.18}$$

$$\nabla \cdot \vec{u} = 0 \tag{2.19}$$

$$\nabla\phi \cdot \vec{n} = \vec{w} \cdot \vec{n} \text{ at } \partial\Omega \tag{2.20}$$

$$\vec{u} \cdot \vec{n} = 0 \text{ at } \partial\Omega \tag{2.21}$$

With this decomposition we can define an orthogonal projection operator $\mathbf{P}$, which maps $\vec{w}$ onto its divergence-free part $\vec{u}$. The projection operator is the following linear operator

$$\mathbf{P} \equiv \mathbf{I} - \mathbf{Q} \tag{2.22}$$

$$\mathbf{Q} \equiv \nabla\Delta^{-1}\nabla\cdot \tag{2.23}$$

where $\mathbf{I}$ is the identity matrix. Notice that

$$\mathbf{P}\vec{u} = \vec{u}, \tag{2.24}$$

$$\mathbf{P}\nabla\phi = 0, \tag{2.25}$$

and

$$\vec{w} = \mathbf{P}\vec{w} + \nabla\phi. \tag{2.26}$$

See [30] for a more rigorous mathematical discussion of the projection operator and the Helmholtz-Hodge decomposition.

For variable-density flows we follow [12] and utilize a density weighted decomposition

$$\vec{w} = \vec{u} + \frac{\nabla\phi}{\rho}. \tag{2.27}$$

The density weighted projection operator is the following linear operator

$$\mathbf{P}_\rho \equiv \mathbf{I} - \mathbf{Q}_\rho \tag{2.28}$$

$$\mathbf{Q}_\rho \equiv \frac{1}{\rho}\nabla L_\rho^{-1}\nabla\cdot \tag{2.29}$$

where $\mathbf{I}$ is the identity matrix and where L_ρ is defined in (2.34). Notice that we still have

$$\mathbf{P}_\rho \vec{u} = \vec{u}, \tag{2.30}$$

$$\mathbf{P}_\rho \frac{\nabla\phi}{\rho} = 0, \tag{2.31}$$

and

$$\vec{w} = \mathbf{P}_\rho \vec{w} + \frac{\nabla\phi}{\rho}. \tag{2.32}$$

In Section 4.2 we will define discrete versions of the projection operator.

2.4 Other Notation

In what follows, we will use the following notation

$$A[\psi] \equiv (\vec{u}\cdot\nabla)\psi, \tag{2.33}$$

and

$$L_\beta[\phi] \equiv \nabla\cdot(\frac{1}{\beta}\nabla\phi). \tag{2.34}$$

For clarity we can expand the forms of (2.33). First the momentum advection terms,

$$A[\vec{u}] \equiv (\vec{u}\cdot\nabla)\vec{u} = \begin{bmatrix} uu_x + vu_y + wu_z \\ uv_x + vv_y + wv_z \\ uw_x + vw_y + ww_z \end{bmatrix} \tag{2.35}$$

then the density advection term,

$$A[\rho] \equiv \vec{u} \cdot \nabla \rho = u\rho_x + v\rho_y + w\rho_z \tag{2.36}$$

then the scalar advection term,

$$A[s] \equiv \vec{u} \cdot \nabla s = us_x + vs_y + ws_z. \tag{2.37}$$

The divergence of the gradient with variable coefficients (2.34) is expanded as

$$L_\beta[\phi] \equiv \nabla \cdot (\frac{1}{\beta}\nabla\phi) = (\frac{1}{\beta}\phi_x)_x + (\frac{1}{\beta}\phi_y)_y + (\frac{1}{\beta}\phi_z)_z \tag{2.38}$$

In the following chapters we shall also use the constant coefficient (where ν is independent of $\vec{x}$) elliptic operator (in vector form) to represent the viscous acceleration terms as follows,

$$L\vec{u} \equiv \nu\Delta\vec{u} = \nu \begin{bmatrix} \Delta u \\ \Delta v \\ \Delta w \end{bmatrix} = \nu \begin{bmatrix} u_{xx} + u_{yy} + u_{zz} \\ v_{xx} + v_{yy} + v_{zz} \\ w_{xx} + w_{yy} + w_{zz} \end{bmatrix} \tag{2.39}$$

Chapter 3

Embedded Boundary Discretization

3.1 Geometric Description

3.1.1 Introduction to Embedded Boundaries

To discretize space we utilize the Embedded Boundary (or cut-cell) method [60, 75, 92, 31]. In the Embedded Boundary (EB) method control volumes are formed by the intersection of the domain with Cartesian grid cells. The intersections can be computed to arbitrary accuracy, and are generated with an $O(N^{\frac{D-1}{D}})$ algorithm (where $N \equiv$ total number of control volumes, including regular) from distance type functions (e.g. a Digital Elevation Model).

With the EB method, for the bulk of the flow, which is $O(N)$ control volumes, we compute on a regular Cartesian grid. We use an EB description for the $O(N^{\frac{D-1}{D}})$ control

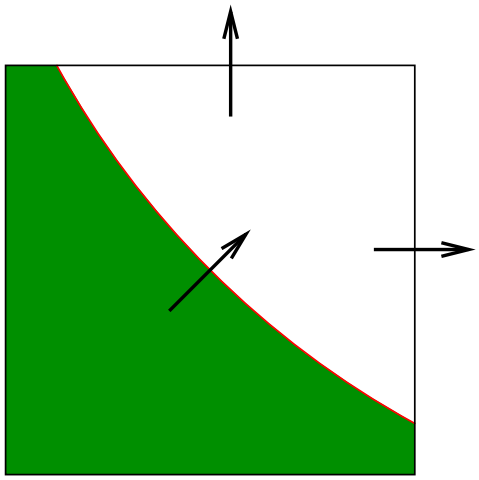

Figure 3.1: Example 2D embedded boundary control volume, arrows indicate fluxes.

volumes that intersect the boundary (where D is the dimensionality of the problem). The advantages of an underlying rectangular grid are as follows: grid generation is tractable, with a straightforward coupling to block-structured adaptive mesh refinement (AMR); good discretization technology, e.g. well-understood consistency theory for conservative finite differences; geometric multigrid for elliptic solvers; away from boundaries, the method reduces to a standard conservative finite difference method.

An example two-dimensional EB is shown in Figure 3.1. Example three-dimensional EB control volumes are shown in Figure 3.2. In these figures the curves indicate the intersection of the exact boundary with a Cartesian cell. Note that to enable second-order accurate methods for this work, we approximate face intersections using quadratic interpolants, but the EB method can be extended to arbitrary accuracy.

To discretize our conservation laws using EB's all we need are the following quantities: volume fractions, area fractions, centroids, boundary areas, and boundary normals.

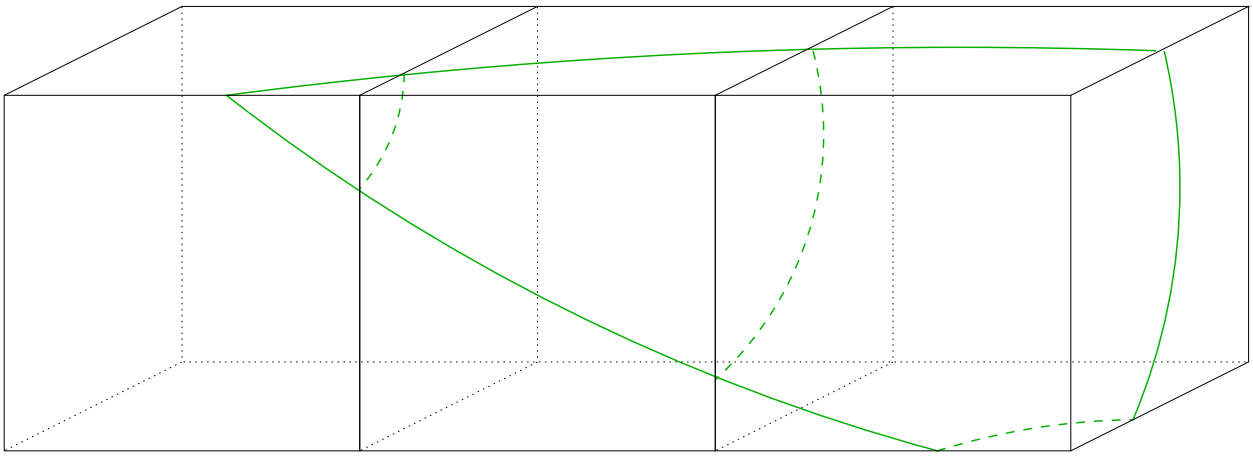

Figure 3.2: Example 3D embedded boundary control volumes.

Given these geometric quantities on a fine grid, we can compute the same quantities on coarser grids without reference to the original geometry (by geometric coarsening). This permits efficient, dynamic coarsening and refinement of highly complex geometry as a simulation progresses (e.g. AMR and multigrid). An example coarsening sequence is shown in Figures 3.3-3.6.

EB's are "water tight" by construction, i.e. if two control-volumes share a face, they both have the same area fraction for that face. Unlike typical complex geometry discretization methods, the EB control volumes naturally fit within easily parallelized disjoint block data structures.

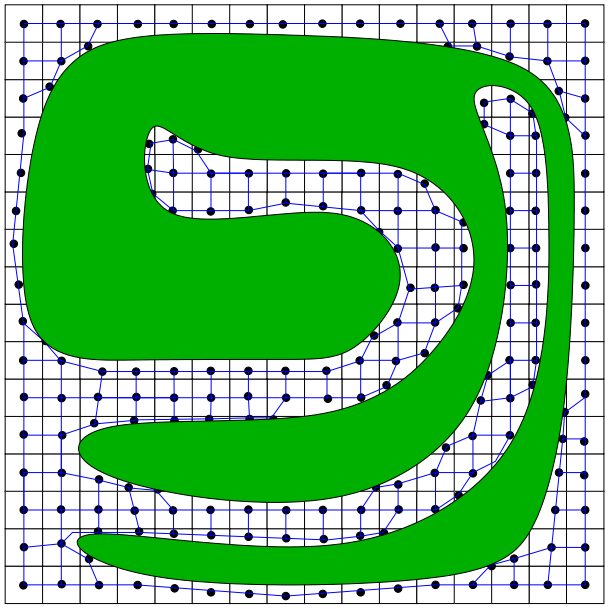

Figure 3.3: An arbitrarily complex object represented on a 16 by 16 grid. On our finest grids the connectivity graph has only one control volume (node) per grid cell.

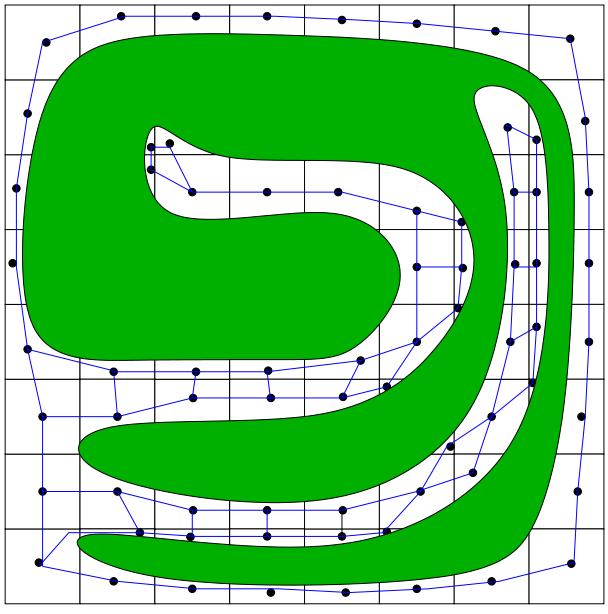

Figure 3.4: The arbitrarily complex object coarsened onto an 8 by 8 grid. The connectivity graph has been coarsened and can contain more than one node per grid cell.

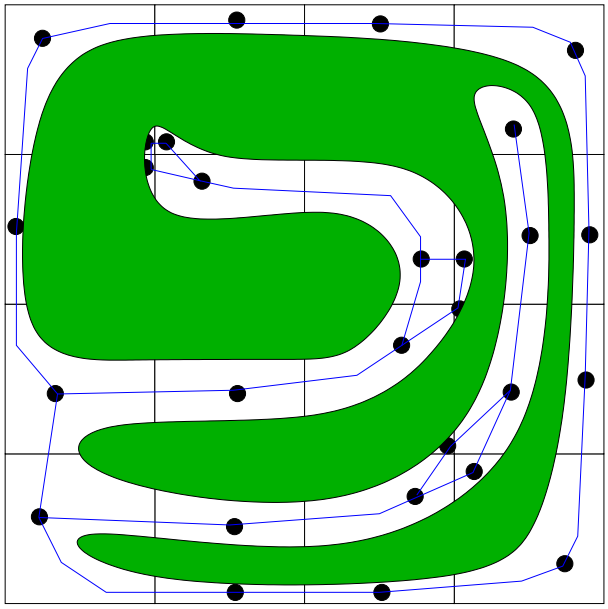

Figure 3.5: The arbitrarily complex object coarsened onto a 4 by 4 grid. The connectivity graph has been coarsened and can contain more than one node per grid cell.

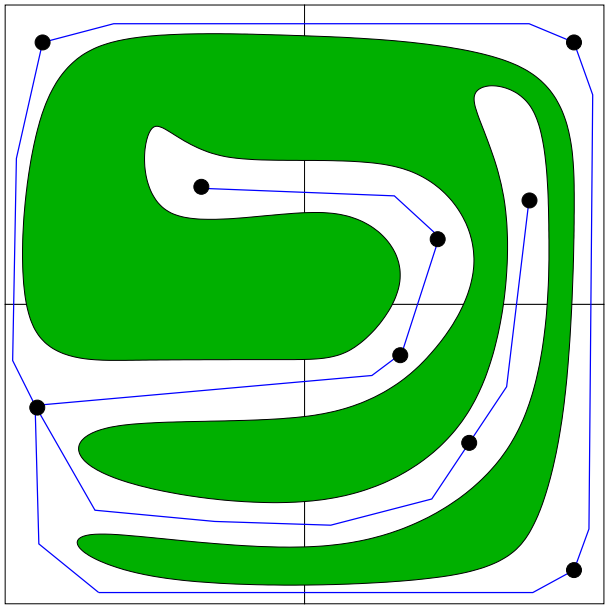

Figure 3.6: The arbitrarily complex object coarsened onto a 2 by 2 grid. The connectivity graph has been coarsened and can contain more than one node per grid cell.

3.1.2 Geometry Generation

Embedded Boundary Notation

The underlying discretization of space is given by rectangular control volumes on a Cartesian grid: $\Upsilon_{\boldsymbol{i}_d} = [\boldsymbol{i}_d h_d, (\boldsymbol{i}_d + \boldsymbol{u}_d)h_d]$, $\boldsymbol{i} \in \mathbb{Z}^D$, where D is the dimensionality of the problem, h_d is the mesh spacing in direction d, and $\boldsymbol{u}$ is the vector whose entries are all ones. In the case of a fixed, irregular domain Ω, the geometry is represented by the intersection of Ω with the Cartesian grid. We obtain control volumes $V_{\boldsymbol{i}} = \Upsilon_{\boldsymbol{i}} \cap \Omega$ and faces $A_{\boldsymbol{i} \pm \frac{1}{2}\boldsymbol{e}_d}$, which are the intersection of $\partial V_{\boldsymbol{i}}$ with the coordinate planes $\{\boldsymbol{x} : x_d = (i_d \pm \frac{1}{2})h_d\}$. Here $\boldsymbol{e}_d$ is the unit vector in the d direction. We also define $A^B_{\boldsymbol{i}}$ to be the intersection of the boundary of the irregular domain with the Cartesian control volume: $A^B_{\boldsymbol{i}} = \partial\Omega \cap \Upsilon_{\boldsymbol{i}}$. We will assume here that there is a one-to-one correspondence between the control volumes and faces and the corresponding geometric entities on the underlying Cartesian grid. The description can be generalized to allow for boundaries whose width is less than the mesh spacing or boundaries with sharp trailing edges.

Geometric Quantities

In order to construct finite difference methods, we will need only a small number of real-valued quantities that are derived from geometric objects.

- Areas and volumes are expressed in these dimensionless terms:

$$\text{Volume fractions: } \kappa_{\boldsymbol{i}} = |V_{\boldsymbol{i}}| \prod_{d=1}^{D} \frac{1}{h_d} \tag{3.1}$$

$$\text{Face apertures: } \alpha_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_d} = |A_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_d}| \prod_{d'\neq d} \frac{1}{h_{d'}} \tag{3.2}$$

$$\text{Boundary apertures: } \alpha_{\boldsymbol{i}}^B = |A_{\boldsymbol{i}}^B| h_1^{-(D-1)} \tag{3.3}$$

We assume that we can compute estimates of the dimensionless quantities that are sufficiently accurate (i.e. $O(h^2)$ for apertures, $O(h)$ for volume fractions).

- The locations of centroids, and the average outward normal to the boundary are given exactly by:

$$\text{Face centroid: } \boldsymbol{x}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = \frac{1}{|A_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}|} \int_{A_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}} \boldsymbol{x} dA \tag{3.4}$$

$$\text{Boundary face centroid: } \boldsymbol{x}_{\boldsymbol{i}}^B = \frac{1}{|A_{\boldsymbol{i}}^B|} \int_{A_{\boldsymbol{i}}^B} \boldsymbol{x} dA \tag{3.5}$$

$$\text{Outward normal: } \boldsymbol{n}_{\boldsymbol{i}}^B = \frac{1}{|A_{\boldsymbol{i}}^B|} \int_{A_{\boldsymbol{i}}^B} \boldsymbol{n}^B dA \tag{3.6}$$

where $\boldsymbol{n}^B$ is the outward normal to $\partial\Omega$, defined for each point on $\partial\Omega$. Again, we assume that we can compute estimates of these quantities that are accurate to $O(h^2)$.

Using just these quantities, we can define conservative discretizations for all the required operators.

3.1.3 Embedded Boundary Examples

Here we present example visualizations to illustrate our ability to approximate complex geometry using the Embedded Boundary method. Our first example is shown in Figure 3.7 and is an approximation of a sphere. It is important to note that in this and the following figures we visualize the EB's as piecewise planar, when in fact we use piecewise quadratic approximations for face intersections. The next example is generated from imaging data of a coral (see Figure 3.8). This coral example illustrates our ability to handle arbitrarily complex geometry in a completely automated and efficient way given a distance function. The next two examples are generated from digital elevation models: San Francisco Bay and Lake Tahoe (see Figures 3.9 and 3.10).

3.2 Divergence of Fluxes

Using just the geometric quantities from Section 3.1.2, we can define conservative discretizations for the divergence operator. Let $\vec{F} = (F^1 \dots F^D)$ be a function of $\boldsymbol{x}$. Using the divergence theorem we have

$$\begin{aligned} \nabla \cdot \vec{F} &\approx \frac{1}{|V_{\boldsymbol{i}}|} \int_{V_{\boldsymbol{i}}} \nabla \cdot \vec{F} dV = \frac{1}{|V_{\boldsymbol{i}}|} \int_{\partial V_{\boldsymbol{i}}} \vec{F} \cdot \boldsymbol{n} dA \\ &\approx \frac{1}{|V_{\boldsymbol{i}}|} \Big[\big(\sum_{\pm=+,-} \sum_{d=1}^{D} \pm |A_{\boldsymbol{i} \pm \frac{1}{2}\boldsymbol{e}^d}| F^d(\boldsymbol{x}_{\boldsymbol{i} \pm \frac{1}{2}\boldsymbol{e}^d}) \big) + |A^B_{\boldsymbol{i}}| \boldsymbol{n}^B_{\boldsymbol{i}} \cdot \vec{F}(\boldsymbol{x}^B_{\boldsymbol{i}}) \Big], \end{aligned} \tag{3.7}$$

where (3.7) is obtained by replacing the integrals of the normal components of the vector field $\vec{F}$ with the values at the face centroids.

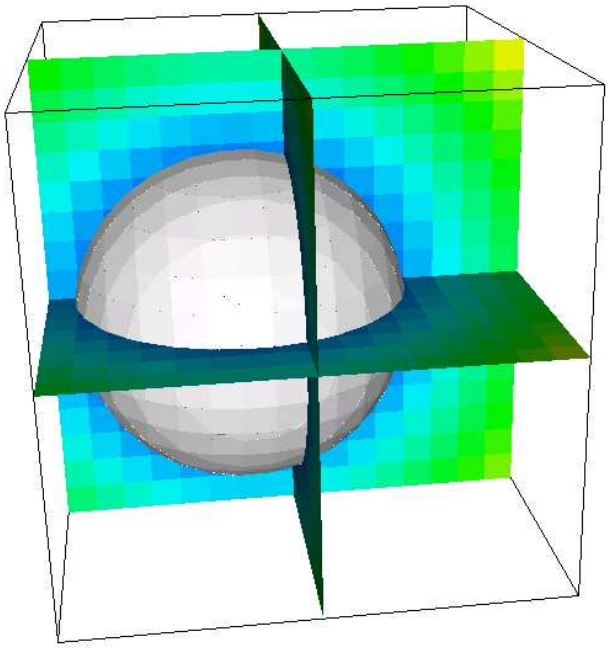

Figure 3.7: EB approximation of a sphere. On the colored slices one can see the edges of individual control volumes.

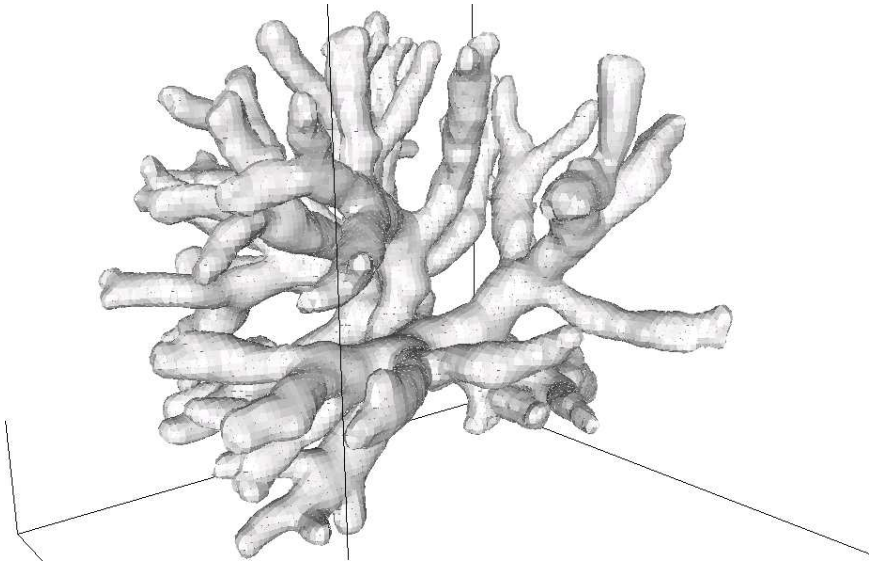

Figure 3.8: EB approximation of a coral. Thanks to Dr. Jaap A. Kaandorp for the Madracis Mirabilis CT scan data.

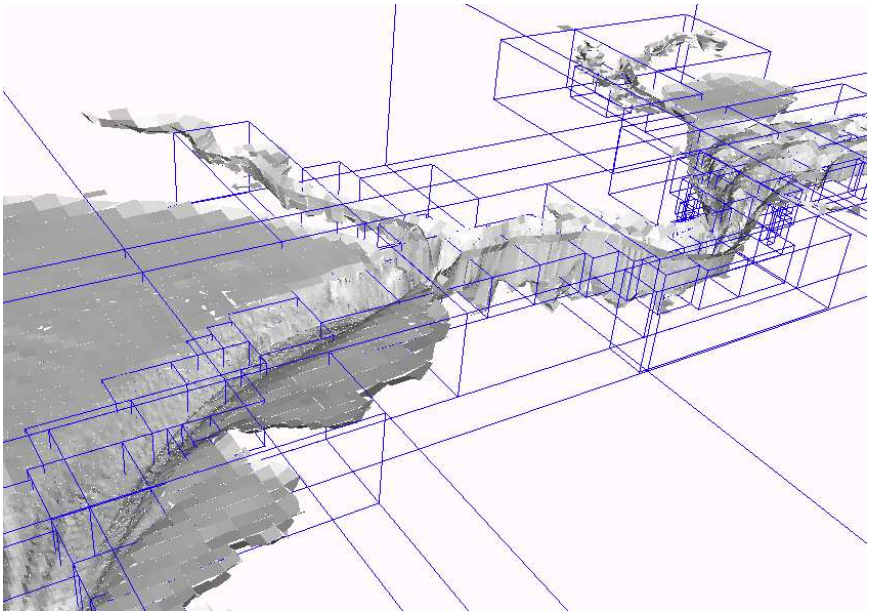

Figure 3.9: EB approximation of Northern San Francisco Bay with adaptive mesh refinement boxes (in blue).

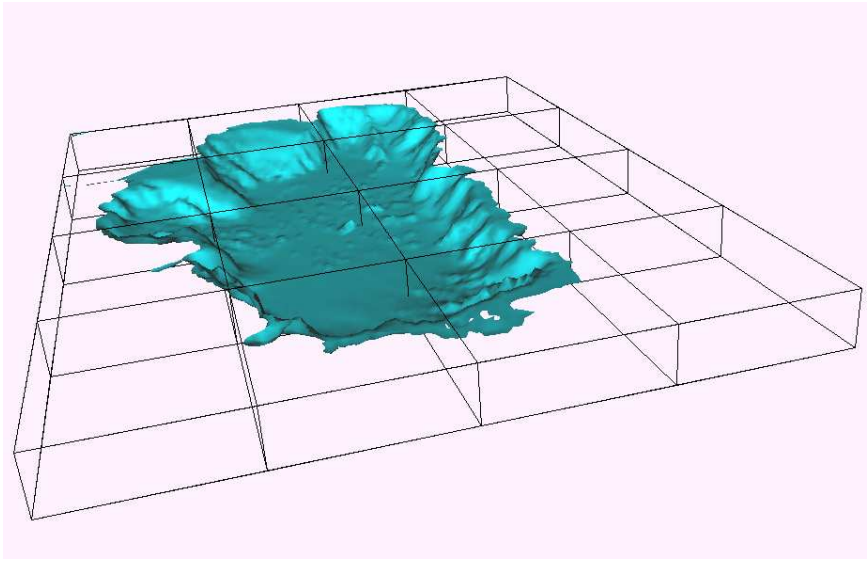

Figure 3.10: EB approximation of Lake Tahoe, with a single level of domain decomposition boxes (in black). Domain decomposition boxes contain control volumes that are within the index space of that box. Thanks to Todd Steissberg for assisting with the Lake Tahoe digital elevation model.

3.3 Elliptic and Parabolic Equations

3.3.1 Poisson's Equation

We first consider Poisson's equation on an irregular domain Ω.

$$\Delta\psi = \rho \text{ on } \Omega$$

$$\frac{\partial\psi}{\partial n} = g_N \text{ on } \partial\Omega \tag{3.8}$$

or

$$\psi = g_D \text{ on } \partial\Omega$$

We define a discrete variable ϕ, $\phi_{\boldsymbol{i}} \approx \psi(\boldsymbol{i}h)$. Using the discretization of the divergence defined in (3.7), we can define a discretization of Poisson's equation ($\nabla \cdot \nabla\phi = \rho$) as follows.

$$(\Delta^h\phi)_{\boldsymbol{i}} = \rho_{\boldsymbol{i}} \tag{3.9}$$

$$(\Delta^h\phi)_{\boldsymbol{i}} = \frac{1}{|V_{\boldsymbol{i}}|}\Big[\big(\sum_{\pm=+,-}\sum_{d=1}^{D} \pm|A_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}^d}|F^d_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}^d}\big) + |A^B_{\boldsymbol{i}}|F^B\Big] \tag{3.10}$$

where $\rho_{\boldsymbol{i}} = \rho(\boldsymbol{i}h)$, and the fluxes F^d and F^B are linear combinations of $\phi_{\boldsymbol{i}}$ and the boundary values ($\vec{F} = \nabla\phi$). In practice, we avoid problems arising from arbitrarily small values of $|V_{\boldsymbol{i}}|$ in the denominator (of 3.10) by recalling (3.1) and solving:

$$\kappa_{\boldsymbol{i}}(\Delta^h\phi)_{\boldsymbol{i}} = \kappa_{\boldsymbol{i}}\rho_{\boldsymbol{i}} \tag{3.11}$$

The fluxes are given by bilinear interpolation of centered differences. Explicitly, bilinear interpolation of fluxes can be written as an iteration of linear interpolation of fluxes

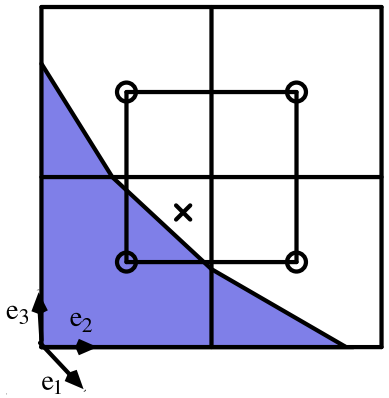

Figure 3.11: Three-dimensional bilinear flux

in the two directions that are not normal to the face. For example, given the face with outward normal $\boldsymbol{e}^1$, with centroid $\boldsymbol{x}$, define the linearly interpolated flux in the d $(d \neq 1)$ direction by

$$F^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^1} = \eta \frac{(\phi_{\boldsymbol{i}+\boldsymbol{e}^1} - \phi_{\boldsymbol{i}})}{h_1} + (1-\eta)\frac{(\phi_{\boldsymbol{i}+\boldsymbol{e}^1 \pm \boldsymbol{e}^d} - \phi_{\boldsymbol{i}\pm\boldsymbol{e}^d})}{h_1}$$

$$\eta = 1 - \frac{|\boldsymbol{x}\cdot\boldsymbol{e}^d|}{h_d} \tag{3.12}$$

$$\pm = \begin{cases} + & \boldsymbol{x}\cdot\boldsymbol{e}^d > 0 \\ - & \boldsymbol{x}\cdot\boldsymbol{e}^d \leq 0. \end{cases}$$

The bilinear interpolation of the flux for the face with normal $\boldsymbol{e}^1$ can be written

$$F_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^1} = \omega F^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^1} + (1-\omega) F^d_{\boldsymbol{i}\pm\boldsymbol{e}^{d'}+\frac{1}{2}\boldsymbol{e}^1}$$

$$\omega = 1 - \frac{|\boldsymbol{x}\cdot\boldsymbol{e}^{d'}|}{h_{d'}} \tag{3.13}$$

$$\pm = \begin{cases} + & \boldsymbol{x}\cdot\boldsymbol{e}^{d'} > 0 \\ - & \boldsymbol{x}\cdot\boldsymbol{e}^{d'} \leq 0 \end{cases}$$

where $d' \neq d$, $d' \neq 1$, as in Figure 3.11.

We note that the particular choice of bilinear interpolation for computing the fluxes is a nontrivial one for obtaining a stable method. In particular, we also tried using simple linear interpolation based on three of the faces in Figure 3.11, omitting the face offset along the diagonal. We found that such a method is unstable for some configurations of adjacent small control volumes, in the sense that point Jacobi fails to converge for any value of the relaxation parameter. No such instability was observed for the bilinear scheme. For that reason, we have chosen to reduce order to a piecewise constant interpolant of the fluxes if all four faces required for a bilinear interpolant are unavailable.

Boundary Conditions

For Neumann boundary conditions the flux on the boundary is specified. For Dirichlet boundary conditions further calculations are necessary. Our higher-order method, for use when the geometry is well resolved, is a generalization of the methods described in [59]. Figure 3.12 shows how this generalizes to 3D.

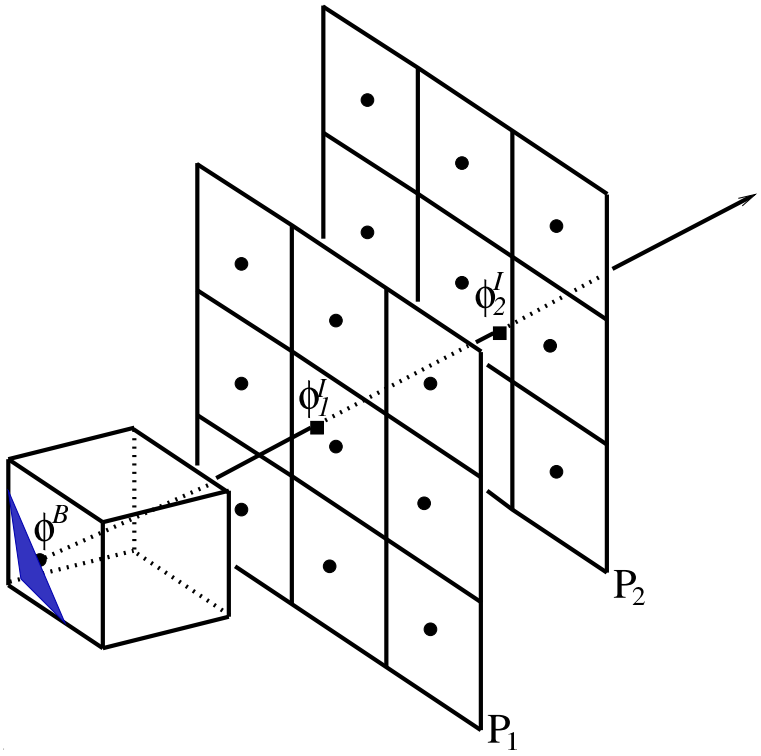

Figure 3.12: Diagram of the second-order stencil for the gradient normal to the interface.

$$\frac{\partial \phi}{\partial n} \approx \frac{1}{d_2 - d_1}\left(\frac{d_2}{d_1}(\phi^B - \phi_1^I) - \frac{d_1}{d_2}(\phi^B - \phi_2^I)\right) \tag{3.14}$$

Here we have used ϕ^B for the value of ϕ on the boundary, which is given by the Dirichlet boundary condition. Interpolation from cell centered values determines ϕ_1^I and ϕ_2^I at distance d_1 and d_2 away from the boundary, respectively.

If all 18 cells are available in Figure 3.12, we make an order $O(h^2)$ estimate $\nabla\phi$ as follows. Depending on the orientation of the normal, two planes are chosen, P_1 and P_2. Using biquadratic interpolation, two values ϕ_1^I and ϕ_2^I are calculated, each requiring 9 values.

The gradient is then calculated by fitting a quadratic to the interpolated values and the value at the interface. We chose the planes P_1, P_2 to be perpendicular to $\boldsymbol{e}^d$, where d is given by

$$\{d : n_d^B \geq n_\ell^B,\ \ell = 1, 2, 3\}. \tag{3.15}$$

In cases where the requisite eighteen cells are not available, we employ a lower-order stencil to estimate the flux to $O(h)$. In 3D this lower-order stencil contains at most eight points including the centroid of the embedded boundary. These eight points are chosen as follows. We associate each one of the eight possible configurations of plus or minus signs of the components of the normal vector with one of the octants in the the coordinate system with origin at the centroid. The stencil consists of the cell-centers of the seven nearest cells in this octant, excluding the cell-center of the control volume containing the boundary centroid itself.

From these points we create an overdetermined linear system to estimate $\nabla\phi$ as follows:

$$\mathcal{A} \cdot \nabla\phi = \delta\phi \tag{3.16}$$

where

$$\mathcal{A} = (\delta\boldsymbol{x}_1, \delta\boldsymbol{x}_2, ..., \delta\boldsymbol{x}_7)^T$$

$$\delta\phi = (\delta\phi_1, \delta\phi_2, ..., \delta\phi_7)^T$$

$$\delta\boldsymbol{x}_m = \boldsymbol{x}_m - \boldsymbol{x}_{\boldsymbol{i}}^B$$

$$\delta\phi_m = \phi_m - \phi_B.$$

We determine $\nabla\phi$ using least squares by solving the normal equations:

$$\mathcal{A}^T\mathcal{A} \cdot \nabla\phi = \mathcal{A}^T\delta\phi.$$

Provided that $\mathcal{A}$ contains three linearly independent rows, $\mathcal{A}^T\mathcal{A}$ is invertible. This is always the case provided the set $\{\boldsymbol{x}_m\}$ contains all points of the form $(\boldsymbol{i} + \boldsymbol{e}^d)h$, plus at least one other point. If that is not the case, we set $\nabla\phi \equiv 0$. These methods lead to a condition number for Δ^h that is bounded independent of κ, and comparable to that of the uniform grid algorithm.

Truncation and Solution Error

We define the truncation error in the usual fashion: $\tau_{\boldsymbol{i}} = \rho_{\boldsymbol{i}} - \Delta^h\phi_{\boldsymbol{i}}^{exact}$, where $\phi_{\boldsymbol{i}}^{exact} = \psi(\boldsymbol{i}h)$. We then have the following asymptotic error estimates for the truncation error. At regular cells

$$\tau_{\boldsymbol{i}} = O(h^2). \tag{3.17}$$

If $\boldsymbol{i}$ is an irregular cell, and the flux on the boundary is second-order accurate, as in (3.14), then

$$\tau_{\boldsymbol{i}} = O\big(\frac{h}{\kappa_{\boldsymbol{i}}}\big). \tag{3.18}$$

If we use the flux computation given by (3.16), the flux on the boundary is first order accurate, and we have

$$\tau_{\boldsymbol{i}} = O\big(\frac{1}{\kappa_{\boldsymbol{i}}}\big). \tag{3.19}$$

We refer to methods that satisfy truncation error estimates of the form (3.18) on the irregular control volumes as being *formally consistent*. We also define the solution error $\epsilon_{\boldsymbol{i}} = \phi_{\boldsymbol{i}} - \phi_{\boldsymbol{i}}^{exact}$.

There is one apparent problem with this truncation error estimate: it is only first order accurate at the boundary. Nonetheless, we observe robust second-order convergence of the solution in max norm [92]. These two facts can be reconciled using a modified equation analysis [59]. Both methods of calculating the gradient for Dirichlet boundary conditions, (3.14) and (3.16), lead to a second-order solution error. This is because, for Dirichlet boundary conditions, solution error is two orders of accuracy more than the truncation error on the boundary.

3.3.2 Variable Coefficient Elliptic Equation

Now we consider a variable coefficient elliptic partial differential equation on an irregular domain Ω. It is this variable coefficient elliptic equation that we use in the density weighted projections (DWPM1 & DWPM2) of Section 4.2.2.

$$\nabla \cdot \beta \nabla \psi = \rho \text{ on } \Omega$$

$$\beta \frac{\partial \psi}{\partial n} = g_N \text{ on } \partial\Omega \tag{3.20}$$

or

$$\psi = g_D \text{ on } \partial\Omega$$

We define a discrete variable ϕ, $\phi_{\boldsymbol{i}} \approx \psi(\boldsymbol{i}h)$. Using the discretization of the divergence defined in (3.7), we can define a discretization of the variable coefficient elliptic equation as follows.

$$L^h[\phi, \beta]_{\boldsymbol{i}} = \rho_{\boldsymbol{i}}, \tag{3.21}$$

$$L^h[\phi, \beta]_{\boldsymbol{i}} = \frac{1}{|V_{\boldsymbol{i}}|}\Big[\big(\sum_{\pm=+,-}\sum_{d=1}^{D} \pm |A_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}^d}| F^d_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}^d}\big) + |A^B_{\boldsymbol{i}}| F^B\Big] \tag{3.22}$$

where $\rho_{\boldsymbol{i}} = \rho(\boldsymbol{i}h)$, and the fluxes F^d and F^B are linear combinations of $\phi_{\boldsymbol{i}}$, $\beta_{\boldsymbol{i}}$, and the boundary values ($\vec{F} = \beta \nabla \phi$). Again, we avoid problems arising from arbitrarily small values of $|V_{\boldsymbol{i}}|$ in the denominator (of 3.22) by recalling (3.1) and solving:

$$\kappa_{\boldsymbol{i}} L^h[\phi, \beta]_{\boldsymbol{i}} = \kappa_{\boldsymbol{i}} \rho_{\boldsymbol{i}} \tag{3.23}$$

The fluxes are given by bilinear interpolation of centered differences. Explicitly, bilinear interpolation of fluxes can be written as an iteration of linear interpolation of fluxes in the two directions that are not normal to the face. For example, given the face with outward normal $\boldsymbol{e}^1$, with centroid $\boldsymbol{x}$, define the linearly interpolated flux in the d $(d \neq 1)$ direction by

$$F^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^1} = \eta\frac{\beta_{\boldsymbol{i}} + \beta_{\boldsymbol{i}+\boldsymbol{e}^1}}{2}\frac{(\phi_{\boldsymbol{i}+\boldsymbol{e}^1} - \phi_{\boldsymbol{i}})}{h_1} + (1-\eta)\frac{\beta_{\boldsymbol{i}+\boldsymbol{e}^1\pm\boldsymbol{e}^d} + \beta_{\boldsymbol{i}\pm\boldsymbol{e}^d}}{2}\frac{(\phi_{\boldsymbol{i}+\boldsymbol{e}^1\pm\boldsymbol{e}^d} - \phi_{\boldsymbol{i}\pm\boldsymbol{e}^d})}{h_1}$$

$$\begin{aligned} \eta &= 1 - \frac{|\boldsymbol{x}\cdot\boldsymbol{e}^d|}{h_d} \\ \pm &= \begin{cases} + & \boldsymbol{x}\cdot\boldsymbol{e}^d > 0 \\ - & \boldsymbol{x}\cdot\boldsymbol{e}^d \le 0. \end{cases} \end{aligned} \tag{3.24}$$

The bilinear interpolation of the flux for the face with normal $\boldsymbol{e}^1$ can be written as in (3.13).

Boundary Conditions

Boundary conditions are as in Section 3.3.1, subject to (3.20).

3.3.3 The Heat Equation

We now consider the heat equation. $\psi : R^3 \times [0,\infty] \to R$ is the unknown and $f : R^3 \times [0,\infty] \to R$ is the source term. We solve

$$\begin{aligned} \frac{\partial\psi}{\partial t} &= \Delta\psi + f \\ \frac{\partial\psi}{\partial n} &= g_N(\boldsymbol{x},t),\ \boldsymbol{x}\in\partial\Omega \\ &\text{or} \\ \psi &= g_D(\boldsymbol{x},t),\ \boldsymbol{x}\in\partial\Omega. \end{aligned} \tag{3.25}$$

We define discrete variables, $\phi_{\boldsymbol{i}}(t), f_{\boldsymbol{i}}(t)$

$$\phi_{\boldsymbol{i}}(t) \approx \psi(\boldsymbol{i}h,t), \quad f_{\boldsymbol{i}}(t) \approx f(\boldsymbol{i}h,t). \tag{3.26}$$

This leads to a semi-discrete system of ODEs

$$\frac{d\phi_{\boldsymbol{i}}}{dt} = \Delta^h \phi_{\boldsymbol{i}} + f_{\boldsymbol{i}} \tag{3.27}$$

We discretize this system in time using the L_0-stable "TGA" method [104], which was also described in [75, 92], as outlined below.

Denote by I the identity operator and by Δ_I^h and Δ_H^h the discrete Laplacian (3.9) with inhomogeneous and homogeneous boundary conditions, respectively. We split the time step Δt such that

$$\begin{aligned} \mu_1 + \mu_2 + \mu_3 &= \Delta t \\ \mu_1 + \mu_2 + \mu_4 &= \Delta t/2. \end{aligned}$$

The update at step n uses the boundary values at the old and new times and also at an intermediate time t_{int}:

$$\begin{aligned} \phi^{n+1} &= (I - \mu_1 \Delta_I^h(t_{new}))^{-1} (I - \mu_2 \Delta_I^h(t_{int}))^{-1} \cdot \\ &\quad [(I + \mu_3 \Delta_I^h(t_{old}))\phi^n + (I + \mu_4 \Delta_H^h) f(t_{avg}) \Delta t] \end{aligned} \tag{3.28}$$

where

$$\begin{aligned} t_{old} &= n\Delta t \\ t_{new} &= (n+1)\Delta t = t_{old} + \mu_1 + \mu_2 + \mu_3 \\ t_{int} &= t_{new} - \mu_1 = t_{old} + \mu_2 + \mu_3 \\ t_{avg} &= (t_{old} + t_{new})/2 = t_{old} + \mu_1 + \mu_2 + \mu_4. \end{aligned}$$

For a second-order L_0-stable method, following [104], we pick $a > 1/2$ and

$$\begin{aligned}
\mu_1 &= \frac{a - \sqrt{a^2 - 4a + 2}}{2}\Delta t \\
\mu_2 &= \frac{a + \sqrt{a^2 - 4a + 2}}{2}\Delta t \\
\mu_3 &= (1 - a)\Delta t \\
\mu_4 &= (\frac{1}{2} - a)\Delta t.
\end{aligned}$$

For a method that uses real arithmetic only, the truncation error is minimized by taking $a = 2 - \sqrt{2} - \epsilon$, where ϵ is machine precision.

It was shown in [75] for Dirichlet boundary conditions that Crank-Nicolson time discretization exhibited oscillatory behavior and furthermore was unstable to some types of forcing at a moving boundary. In [61] this behavior was attributed to the combination of the neutral stability of Crank-Nicolson at high wave numbers and the presence of eigenvalues of Δ^h with non-trivial imaginary parts, corresponding to eigenvalues with oscillatory behavior near the boundary.

3.4 Hyperbolic Equations

We use the explicit hyperbolic methodology developed in [31, 83], with a few minor deviations for incompressible flow. We describe this in detail in Section (4.3).

Chapter 4

The Incompressible Navier-Stokes Equations on a Single Grid

In this chapter we describe the single grid algorithm and the discretizations necessary to solve the incompressible Navier-Stokes equations in this work.

4.1 Semidiscrete Version of Projection Formulation

Using the hyperbolic methodology of Section 3.4, the elliptic and parabolic methodology of Section 3.3.1, and the projection ideas from Section 2.3 we are now in a position to describe our solution strategy for the incompressible Navier-Stokes equations. The overall algorithm for advancing a single time step is presented sequentially in the following Sub-Sections.

4.1.1 Compute Advective Terms

Compute the cell-centered advective terms (2.35), (2.36), and (2.37): $A[\vec{u}]^{n+\frac{1}{2}}$, $A[\rho]^{n+\frac{1}{2}}$, and $A[s]^{n+\frac{1}{2}}$. See Section 4.3 for details.

4.1.2 Update Scalars

We advect ρ with,

$$\frac{\rho^{n+1}-\rho^n}{\Delta t} + A[\rho]^{n+\frac{1}{2}} = 0. \tag{4.1}$$

We compute half time ρ as follows

$$\rho^{n+\frac{1}{2}} = \frac{\rho^{n+1}+\rho^n}{2}. \tag{4.2}$$

The discrete passive scalar update is,

$$s^{n+1} = (I-\mu_1 L)^{-1}(I-\mu_2 L)^{-1}\left[(I+\mu_3 L)s^n + (I+\mu_4 L)f_s^{n+\frac{1}{2}}\Delta t\right] \tag{4.3}$$

where,

$$f_s^{n+\frac{1}{2}} \equiv -A[s]^{n+\frac{1}{2}} + H_s^{n+\frac{1}{2}}. \tag{4.4}$$

Note that for simplicity in (4.3) the Laplacian operators (L) include the diffusion constant (k_s), as in (2.39).

4.1.3 Predict Velocity

We predict the new velocity by solving the parabolic system for the cell-centered $\vec{u}^*$ using TGA from Section 3.3.3. The parabolic (using TGA) time discretized momentum equations are,

$$\vec{u}^{n+1} = (I-\mu_1 L)^{-1}(I-\mu_2 L)^{-1}\left[(I+\mu_3 L)\vec{u}^n + (I+\mu_4 L)f^{n+\frac{1}{2}}\Delta t\right] \tag{4.5}$$

where,

$$f^{n+\frac{1}{2}} \equiv -A[\vec{u}]^{n+\frac{1}{2}} - \frac{\nabla p^{n+\frac{1}{2}}}{\rho^{n+\frac{1}{2}}} + \vec{g}^{n+\frac{1}{2}}. \tag{4.6}$$

$A[\vec{u}]$ is defined in (2.35), and it's discretization is defined in Section 4.3. Since we don't know the new gradient of pressure and can't ensure a divergence-free $\vec{u}^{n+1}$, we take a predictor (or fractional) step for $\vec{u}^*$ (using TGA),

$$\vec{u}^* = (I - \mu_1 L)^{-1}(I - \mu_2 L)^{-1}\left[(I + \mu_3 L)\vec{u}^n + (I + \mu_4 L)f^{n-\frac{1}{2}}\Delta t\right] \tag{4.7}$$

where,

$$f^{n-\frac{1}{2}} \equiv -A[\vec{u}]^{n+\frac{1}{2}} - \frac{\nabla p^{n-\frac{1}{2}}}{\rho^{n+\frac{1}{2}}} + \vec{g}^{n+\frac{1}{2}}. \tag{4.8}$$

Note that for simplicity in (4.5) and (4.7) the Laplacian operators (L) include the diffusion constant (ν), as in (2.39).

4.1.4 Correct Predicted Velocity

We correct $\vec{u}^*$ by approximately projecting it onto a divergence-free space. We do this by using $\mathbf{P}^{cc}_{\rho}$, a cell-centered discretization of the projection operator, and update the pressure gradient. Following [2, 9, 12, 64] and (4.5)-(4.8) we have

$$\vec{u}^* = \vec{u}^{n+1} + \Delta t \frac{\nabla p^{n+\frac{1}{2}} - \nabla p^{n-\frac{1}{2}}}{\rho^{n+\frac{1}{2}}} + O(\Delta t^3). \tag{4.9}$$

We present two different density weighted approximate projection formulations. The first method we term DWPM1 and is given by

$$\vec{u}^{n+1} = \mathbf{P}^{cc}_{\rho^{n+\frac{1}{2}}}\left(\vec{u}^* + \Delta t\big(\frac{\nabla p^{n-\frac{1}{2}}}{\rho^{n+\frac{1}{2}}} - \vec{g}\big)\right) + \Delta t\vec{g}, \tag{4.10}$$

$$\frac{\nabla p^{n+\frac{1}{2}}}{\rho^{n+\frac{1}{2}}} = \frac{1}{\Delta t}\mathbf{Q}^{cc}_{\rho^{n+\frac{1}{2}}}\left(\vec{u}^* + \Delta t\big(\frac{\nabla p^{n-\frac{1}{2}}}{\rho^{n+\frac{1}{2}}} - \vec{g}\big)\right). \tag{4.11}$$

The second method we term DWPM2 and is given by

$$\vec{u}^{n+1} = \mathbf{P}^{cc}_{\rho^{n+\frac{1}{2}}}\left(\vec{u}^{*}\right), \tag{4.12}$$

$$\frac{\nabla p^{n+\frac{1}{2}}}{\rho^{n+\frac{1}{2}}} = \frac{\nabla p^{n-\frac{1}{2}}}{\rho^{n+\frac{1}{2}}} + \frac{1}{\Delta t}\mathbf{Q}^{cc}_{\rho^{n+\frac{1}{2}}}\left(\vec{u}^{*}\right). \tag{4.13}$$

These approximate cell-centered projections differ in that the first method, DWPM1, we solve an elliptic equation for the pressure ($p^{n+\frac{1}{2}}$), while in the second method, DWPM2, we solve an elliptic equation for the pressure change ($p^{n+\frac{1}{2}} - p^{n-\frac{1}{2}}$). With DWPM1 we have the benefit of an implicit solve for pressure at every time step (at $n+\frac{1}{2}$). The DWPM1 drawback is that it is not clear how to extend the pressure outflow boundary conditions to variable-density outflows in hydrostatic balance. With DWPM2 we have desirable $p_t = 0$ outflow boundary conditions, but we have an incremental update for the pressure ($p^{n+\frac{1}{2}}$) at every time step. A secondary drawback to DWPM2 is that the pressure and pressure gradient is subject to accumulation of high frequency error for flows where the pressure is nearly constant in time. This is due to the fact that there is no mechanism to damp these errors with an incremental update (DWPM1 damps the errors via the elliptic solve for $p^{n+\frac{1}{2}}$), see [2] for a relevant discussion. Nonetheless, DWPM2 produces good results as is seen in Section 6.5.2.

As part of the cell-centered projection, our last step is to apply a filter to damp divergent modes that are in the null space of our cell-centered projection (see Section 4.2.2).

4.2 Discretizing Projections

In this work we need to define projections for two data types: face-centered and cell-centered. The face-centered operator $\mathbf{P}^{mac}_{\rho}$ is defined in Section 4.2.1, while the cell-

centered projection operator $\mathbf{P}_\rho^{cc}$ is defined in Section 4.2.2.

In Sections 4.2.1 and 4.2.2, velocity boundary conditions for the velocity operators are specified in Section 2.2.1, and pressure boundary conditions for the pressure operators are specified in Section 2.2.2. Note that the pressure difference $(p^{n+\frac{1}{2}} - p^{n-\frac{1}{2}})$ elliptic solves (in DWPM2) have a desirable homogeneous boundary condition at outflow, eliminating the need to approximate a variable-density hydrostatic pressure (as is the case for DWPM1 at outflow).

4.2.1 Face-Centered MAC Projection

This projection is based on face-centered advective velocities. The Hodge decomposition of the advective velocities is

$$u^{*,d}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = u^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} + [\frac{\nabla^d \phi}{\rho}]_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}. \tag{4.14}$$

We first define the discrete operators needed for the projection, then we define the face-centered projection operator.

Face-Centered Divergence

We define face-centered vector fields $\vec{F} = (F_1, ..., F_{\mathbf{D}})$, such that $\vec{F}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$ is at face-centers. For regular control-volumes (non-EB) we define a discretized divergence operator on such a vector field as follows

$$D^{mac} \cdot \vec{F} \equiv \frac{1}{|V_{\boldsymbol{i}}|}\Big[\big(\sum_{\pm=+,-}\sum_{d=1}^{D} \pm |A_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}^d}| F^d(\boldsymbol{x}_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}^d})\big)\Big], \tag{4.15}$$

where (4.15) is obtained by replacing the integrals of the normal components of the vector field $\vec{F}$ with the values at face centers. For irregular control volumes (non-EB) we define a

discretized divergence operator on such a vector field as follows

$$D^{mac} \cdot \vec{F} \equiv \frac{1}{|V_{\boldsymbol{i}}|}\Big[\big(\sum_{\pm=+,-}\sum_{d=1}^{D} \pm |A_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}^d}| F^d(\boldsymbol{x}_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}^d})\big) + |A_{\boldsymbol{i}}^B| \boldsymbol{n}_{\boldsymbol{i}}^B \cdot \vec{F}(\boldsymbol{x}_{\boldsymbol{i}}^B)\Big], \tag{4.16}$$

where (4.16) is obtained by replacing the integrals of the normal components of the vector field $\vec{F}$ with the values at the face centroids.

Face-Centered Gradient

We define the face-centered gradient $G^{mac,d}[\phi]_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$ for each direction d using

$$G^{mac,d}[\psi]_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = \frac{\psi_{\boldsymbol{i}+\boldsymbol{e}} - \psi_{\boldsymbol{i}}}{\Delta x_d}. \tag{4.17}$$

To scale the MAC-gradient by density ($\rho_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$) we divide as follows

$$[\frac{G^{mac,d}\phi}{\rho}]_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = (\phi_{x_d,\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d})/(\frac{\rho_{\boldsymbol{i}} + \rho_{\boldsymbol{i}+\boldsymbol{e}_d}}{2}). \tag{4.18}$$

For irregular faces, we use an interpolation operator $I_{\mathbf{fc}}^{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$ to move data from face-centers to face-centroids, see (3.12) and (3.13).

MAC-Projection

Our face-centered MAC projection operator does the following

$$u^{*,d}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = \mathbf{P}^{mac}_{\rho}\big(u^{*,d}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}\big). \tag{4.19}$$

We use the face-centered, discrete operators to define the MAC-projection

$$\mathbf{P}^{mac}_{\rho} \equiv \mathbf{I} - \frac{1}{\rho}G^{mac}[D^{mac} \cdot \frac{1}{\rho}G^{mac}]^{-1}D^{mac}. \tag{4.20}$$

The first step is to compute the MAC-divergence of (4.14) and then solve a variable coefficient (see Section 3.3.2) elliptic equation for $\phi_{\boldsymbol{i}}$ given boundary conditions on ϕ (see Section

2.2.2),

$$D^{mac} \cdot [\frac{1}{\rho_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}} G_d^{mac} \phi_{\boldsymbol{i}}] = D^{mac} \cdot [u^{*,d}_{\boldsymbol{i}+\frac{\mathbf{1}}{\mathbf{2}}\mathbf{e_d}}]. \tag{4.21}$$

We subsequently solve for the divergence-free advective velocities

$$u^d_{\boldsymbol{i}+\frac{\mathbf{1}}{\mathbf{2}}\mathbf{e_d}} = u^{*,d}_{\boldsymbol{i}+\frac{\mathbf{1}}{\mathbf{2}}\mathbf{e_d}} - [\frac{G^{mac,d}\phi}{\rho}]_{\boldsymbol{i}+\frac{\mathbf{1}}{\mathbf{2}}\mathbf{e_d}}. \tag{4.22}$$

4.2.2 Cell-Centered Projections

Now our Hodge decomposition of the cell-centered velocities is

$$\vec{u}^*_{\boldsymbol{i}} = \vec{u}_{\boldsymbol{i}} + [\frac{\nabla\phi}{\rho}]_{\boldsymbol{i}}. \tag{4.23}$$

Below we define the cell-centered discretizations needed to define our cell-centered projection operator.

Average Cell to Face

In what follows we need to interpolate the cell-centered $\vec{w}^*_{\boldsymbol{i}}$ (advective components only) using a second-order accurate interpolant to face centers (A), then second-order to face centroids (I),

$$w^{*,d}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = I^{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}_{\mathbf{fc}}(A^{C\to F}(\vec{w}^*)) \tag{4.24}$$

So we need to define our averaging operator to move data from cell-centers to face-centers

$$[A^{C\to F}\psi]_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} \equiv \frac{\psi_{\boldsymbol{i}} + \psi_{\boldsymbol{i}+\boldsymbol{e}_d}}{2}. \tag{4.25}$$

Cell-Centered Gradient

In what follows we also need to define $G^{cc}[\psi]_{\boldsymbol{i}}$. For component d, the cell-centered gradient is

$$G_d^{cc}[\psi]_{\boldsymbol{i}} = \frac{G_d^{mac}[\psi]_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} + G_d^{mac}[\psi]_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}_d}}{2}, \tag{4.26}$$

where the face-centered gradients for un-covered faces are computed by (4.17). For covered faces, we extrapolate to the covered face center by using uncovered values, i.e. (4.17). For regular control volumes this procedure reduces to a traditional centered difference.

Cell-Centered Projection

Here we are solving a similar problem as in the MAC case. To do this, we average cell-centered velocities to faces, apply the MAC projection, and average the gradient field back to cell centers. Our cell-centered projection operator does the following

$$\vec{u}_{\boldsymbol{i}} = \mathbf{P}_\rho^{cc}\big(\vec{w}_{\boldsymbol{i}}^*\big). \tag{4.27}$$

Now we define our cell-centered projection.

$$\mathbf{P}_\rho^{cc} \equiv \mathbf{I} - \mathbf{Q}_\rho^{cc} \tag{4.28}$$

$$\mathbf{Q}_\rho^{cc} \equiv \frac{1}{\rho} G^{cc}[D^{mac} \cdot \frac{1}{\rho} G^{mac}]^{-1} \Big[D^{mac} \cdot A^{C \to F} \Big] \tag{4.29}$$

and where $A^{C \to F}$ is an averaging operator that moves data from cell-centers (C) to face-centers (F).

Note, that the cell-centered projection uses a MAC-projection to obtain the face-centered gradient, which we then use to compute G^{cc}. Also note that we use homogeneous

solid-wall boundary conditions for D^{mac}. We use homogeneous Neumann boundary conditions for ϕ on solid-wall boundaries. These boundary conditions lead to a consistent discretization of (2.17)-(2.21), as discussed in [64].

Filtering Divergent Velocity Modes

Since our cell-centered approximate projection methods (see Section 4.2.2) have a null space where divergent velocity fields can exist [89], we damp those modes with a divergence sensitive filter:

$$\vec{u}_{\boldsymbol{i}}^{n+1} := \vec{u}_{\boldsymbol{i}}^{n+1} + \lambda GD\vec{u}_{\boldsymbol{i}}^{n+1} \tag{4.30}$$

We discretize the divergence D using a face-centered discretization, and the gradient G using a cell-centered discretization, with λ as a damping coefficient. GD is an approximation of the matrix-valued operator $\partial_{x_i}\partial_{x_j}$. This is algebraically equivalent to a projection in which, instead of solving an elliptic equation, only one Jacobi iteration is taken toward the solution from a zero starting point.

4.3 Advection Details

Here we present a second-order accurate and stable Godunov methodology for computing discrete approximations to advective terms: (2.35), (2.36), and (2.37). This is an extension of [31] to the incompressible Navier-Stokes equations. To do this we:

1. Extrapolate the advective normal velocities to face centers at $t^{n+\frac{1}{2}}$
2. MAC-project the advective normal velocities to obtain divergence-free advective velocities

3. Extrapolate the remaining tangential velocities, ρ and s to face centers at $t^{n+\frac{1}{2}}$

4. Compute the advective terms as a combination of stable and conservative approximations

4.3.1 Extrapolate Advective Velocities to Face Centers at $n+\frac{1}{2}$

We describe the Godunov methodology used to compute second-order accurate approximations to the advective velocities. The method is a characteristic extrapolation procedure. For a given scalar s we extrapolate, in direction d, to the low (L) side of the $\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d$ face center and at time $n+\frac{1}{2}$, by

$$\tilde{s}^{L,n+\frac{1}{2}}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = s^n_{\boldsymbol{i}} + \frac{\Delta x_d}{2}\frac{\partial s}{\partial x_d}\bigg|^n_{\boldsymbol{i}} + \frac{\Delta t}{2}\frac{\partial s}{\partial t}\bigg|^n_{\boldsymbol{i}}. \tag{4.31}$$

This is a Taylor expansion in time and space. Similarly for component c of the velocity we extrapolate in the d-direction to the low (L) side of the $\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d$ face center,

$$\tilde{u}^{L,n+\frac{1}{2}}_{d',\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = u^n_{d',\boldsymbol{i}} + \frac{\Delta x_d}{2}\frac{\partial u_{d'}}{\partial x_d}\bigg|^n_{\boldsymbol{i}} + \frac{\Delta t}{2}\frac{\partial u_{d'}}{\partial t}\bigg|^n_{\boldsymbol{i}}, \tag{4.32}$$

and on the high (H) side of the $\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}_d$ face center,

$$\tilde{u}^{H,n+\frac{1}{2}}_{d',\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}_d} = u^n_{d',\boldsymbol{i}} - \frac{\Delta x_d}{2}\frac{\partial u_{d'}}{\partial x_d}\bigg|^n_{\boldsymbol{i}} + \frac{\Delta t}{2}\frac{\partial u_{d'}}{\partial t}\bigg|^n_{\boldsymbol{i}}. \tag{4.33}$$

Since we are extrapolating advective components only, we evaluate (4.32) and (4.33) with $d'=d$. We approximate spatial derivatives in direction d by,

$$\frac{\partial u_d}{\partial x_d}\bigg|^n_{\boldsymbol{i}} \approx \left(u_d\right)^{lim,4}_{x_d,\boldsymbol{i}}, \tag{4.34}$$

where the limited slopes are defined in Section 4.3.4. For these advective component extrapolations we define preliminary advective velocities by interpolating to face centers

$$\vec{u}^{pre}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = I^{\boldsymbol{i}}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}\vec{u}^n_{\boldsymbol{i}}. \tag{4.35}$$

We then approximate the time derivative term for each direction d,

$$\left.\frac{\partial u_d}{\partial t}\right|_{\boldsymbol{i}}^{n} \approx H_{\boldsymbol{i}}^{n} - \sum_{d=1}^{D} u_{\boldsymbol{i}}^{d,pre} \left(u_d\right)_{x_d,\boldsymbol{i}}^{lim,4}. \tag{4.36}$$

Where the transverse derivatives are upwinded, and

$$u_{\boldsymbol{i}}^{d,ave} = \frac{1}{2}(u_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}^{d,pre} + u_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}_d}^{d,pre}), \tag{4.37}$$

and we define the source term of the dth component,

$$H_{\boldsymbol{i}}^{n,d} = \frac{1}{\rho^n} L^d[\vec{u}^n, \mu]_{\boldsymbol{i}} - [\frac{\nabla^d p}{\rho}]_{\boldsymbol{i}}^{n-\frac{1}{2}} + \vec{g}_{\boldsymbol{i}}^{d,n}. \tag{4.38}$$

This yields the advective velocities on the low and high side of face centers. For covered faces in irregular cells we extrapolate both the extrapolations, (4.32) and (4.33), and the preliminary advective velocities (4.35) to covered faces [31].

Now we solve the Riemann problem by upwinding to choose the state for each direction d advective velocity,

$$\tilde{u}_{d,\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}^{n+\frac{1}{2}} = R(\tilde{u}_{d,\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}^{L,n+\frac{1}{2}}, \tilde{u}_{d,\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}^{H,n+\frac{1}{2}}, u_{d,\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}^{pre}, d), \tag{4.39}$$

where R is defined in (4.59).

We MAC-project $\vec{\tilde{u}}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}^{n+\frac{1}{2}}$, this results in a divergence-free $\vec{u}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}^{ADV}$ advective velocity field

$$\vec{u}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}^{ADV} = \mathbf{P}_{\rho}^{mac}(\tilde{u}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}^{n+\frac{1}{2},d}). \tag{4.40}$$

4.3.2 Extrapolate Remaining Quantities to Face Centers at $n + \frac{1}{2}$

We are now in a position to extrapolate the remaining quantities to half time at face centers. We do this similarly to how we extrapolated the advective velocities. But,

now we use the divergence-free advective velocities at face centroids, with a non-conservative (but stable) discretization of the advective terms in the time derivative approximation. Here we present the method for a generic scalar s, which will represent the tangential velocities, or scalars that we are advecting with the flow. We approximate the spatial derivatives in (4.31) using our limited spatial derivatives method, see (4.57),

$$\left.\frac{\partial s}{\partial x_d}\right|_{\boldsymbol{i}}^{n} \approx s_{x_d,\boldsymbol{i}}^{lim,4}. \tag{4.41}$$

For the time derivative we use the divergence-free face-centered advective velocities

$$\left.\frac{\partial s}{\partial t}\right|_{\boldsymbol{i}}^{n} \approx H_{\boldsymbol{i}}^{n} - \sum_{d=1}^{D} u_{\boldsymbol{i}}^{d,ave} s_{x_d,\boldsymbol{i}}^{lim,4}. \tag{4.42}$$

Where the transverse derivatives are upwinded, and

$$u_{\boldsymbol{i}}^{d,ave} = \frac{1}{2}(u_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}^{d,ADV} + u_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}_d}^{d,ADV}), \tag{4.43}$$

and we define the source term,

$$H_{\boldsymbol{i}}^{n} = L[s^n, k_s]_{\boldsymbol{i}} + H_{s,\boldsymbol{i}}^{n}. \tag{4.44}$$

For covered faces in irregular cells we extrapolate both the extrapolations, (4.31), high or low, depending on what is needed), and the advective velocities to covered faces using the method of [31].

Subsequently we solve a Riemann problem using the divergence-free face-centered velocities $\vec{u}_{\boldsymbol{i}+\frac{\mathbf{1}}{\mathbf{2}}\boldsymbol{e}_\mathbf{d}}^{ADV}$,

$$\tilde{s}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}^{n+\frac{1}{2}} = R(\tilde{s}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}^{L,n+\frac{1}{2}}, \tilde{s}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}^{H,n+\frac{1}{2}}, \left(u_d\right)_{\boldsymbol{i}+\frac{\mathbf{1}}{\mathbf{2}}\boldsymbol{e}_\mathbf{d}}^{ADV}, d), \tag{4.45}$$

where R is defined in (4.59). For tangential velocities at face centers (including covered faces) we enforce a divergence-free condition by correcting with the mac-gradients ($u^{d'}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = u^{*,d'}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} - [\frac{\nabla^{d'}\phi}{\rho^{n+\frac{1}{2}}}]_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$). Now we have $\tilde{s}^{n+\frac{1}{2}}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$ at regular, irregular, and needed covered face centers.

4.3.3 Compute Advective Terms

Since the advective velocities are exactly discretely divergence-free, i.e.

$$D^{mac} \cdot \vec{u} = 0, \tag{4.46}$$

then the discrete analog of

$$(\vec{u} \cdot \nabla)c = \nabla \cdot (c\vec{u}) = \nabla \cdot \vec{F} \tag{4.47}$$

is true. We will proceed by selectively using (4.47) to conservatively discretize the advective terms (2.35), (2.36), and (2.37). Following [31], we compute a stable (S) hybridization of conservative (C) and non-conservative (NC) divergences,

$$(D \cdot \vec{F})^{S,NC}_{\boldsymbol{i}} = \kappa_{\boldsymbol{i}}(D \cdot \vec{F})^{C}_{\boldsymbol{i}} + (1 - \kappa_{\boldsymbol{i}})(D \cdot \vec{F})^{NC}_{\boldsymbol{i}} \tag{4.48}$$

where

$$(D \cdot \vec{F})^{NC}_{\boldsymbol{i}} = \frac{1}{\Delta x_d} \sum_{\pm=+,-} \sum_{d=1}^{D} \pm \vec{F}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} \tag{4.49}$$

and where $(D \cdot \vec{F})^{C}_{\boldsymbol{i}}$ is defined below. Note that for (4.49) we use covered face values for faces with zero apertures. The covered face values are obtained by extrapolating from the interior as in [31].

For velocity at irregular control volumes and for scalars at all control volumes we define $(D \cdot \vec{F})^{C}_{\boldsymbol{i}}$ using the flux based divergence of (3.7). For velocity component d' at regular

control volumes we define $(D \cdot \vec{F})^C_{\boldsymbol{i}}$ as follows:

$$(D \cdot \vec{F})^C_{\boldsymbol{i}} = \sum_{d=1}^{D} \left[\frac{\vec{u}_{d,\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} + \vec{u}_{d,\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d}}{2} \left(\frac{\vec{u}_{d',\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - \vec{u}_{d',\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d}}{\Delta x_d} \right) \right]. \tag{4.50}$$

Notice that κ in the denominator of (3.7) cancels with the κ of (4.48), eliminating small κ division issues.

A non-stable (NS) but conservative update for density would be

$$\rho^{n+1,NS,C}_{\boldsymbol{i}} = \rho^n_{\boldsymbol{i}} - \Delta t (D \cdot \vec{F})^C_{\boldsymbol{i}}, \tag{4.51}$$

while a stable but non-conservative update would be

$$\rho^{n+1,S,NC}_{\boldsymbol{i}} = \rho^n_{\boldsymbol{i}} - \Delta t (D \cdot \vec{F})^{S,NC}_{\boldsymbol{i}}. \tag{4.52}$$

We would like to update with (4.52), but we need to account for the missing mass. The missing mass is (4.51) minus (4.52) scaled by the volume, and using (4.48) is

$$\delta M_{\boldsymbol{i}} = \Delta t |V_{\boldsymbol{i}}| (1 - \kappa_{\boldsymbol{i}}) \left[(D \cdot \vec{F})^{NC}_{\boldsymbol{i}} - (D \cdot \vec{F})^C_{\boldsymbol{i}} \right]. \tag{4.53}$$

To enforce exact mass conservation, we use volume weighted redistribution of this mass to neighboring control volumes

$$\rho^{n+1,S,C}_{\boldsymbol{i}'} = \rho^n_{\boldsymbol{i}'} - \Delta t (D \cdot \vec{F})^{S,NC}_{\boldsymbol{i}'} + \frac{\kappa_{\boldsymbol{i}'}}{\sum_{\boldsymbol{i}'' \in N(\boldsymbol{i})} \kappa_{\boldsymbol{i}''}} \frac{\delta M_{\boldsymbol{i}}}{|V_{\boldsymbol{i}'}|} \tag{4.54}$$

or

$$(D \cdot \vec{F})^{S,C}_{\boldsymbol{i}'} = (D \cdot \vec{F})^{S,NC}_{\boldsymbol{i}'} - \frac{\kappa_{\boldsymbol{i}'}}{\sum_{\boldsymbol{i}'' \in N(\boldsymbol{i})} \kappa_{\boldsymbol{i}''}} \frac{\delta M_{\boldsymbol{i}}}{|V_{\boldsymbol{i}'}| \Delta t}. \tag{4.55}$$

More generally (for velocity too),

$$(D \cdot \vec{F})^{S,C}_{\boldsymbol{i}'} = (D \cdot \vec{F})^{S,NC}_{\boldsymbol{i}'} - \frac{1}{\sum_{\boldsymbol{i}'' \in N(\boldsymbol{i})} \kappa_{\boldsymbol{i}''}} \kappa_{\boldsymbol{i}} (1 - \kappa_{\boldsymbol{i}}) \left[(D \cdot \vec{F})^{NC}_{\boldsymbol{i}} - (D \cdot \vec{F})^C_{\boldsymbol{i}} \right]. \tag{4.56}$$

Where $\boldsymbol{i}' \in N(\boldsymbol{i})$, and where $N(\boldsymbol{i})$ is a set of indices whose components differ from those of $\boldsymbol{i}$ by no more than one and can be reached by a monotonic path.

4.3.4 Limited Slope Computation

Here we define the limited slope operator in direction d, used in (4.34), (4.36), (4.41), and (4.42). The second-order accurate slope calculation is

$$\psi^{lim,2}_{x_d,\boldsymbol{i}} = \begin{cases} \frac{sign(\psi_{\boldsymbol{i}+\boldsymbol{e}_d}-\psi_{\boldsymbol{i}-\boldsymbol{e}_d})min(2|\psi_{\boldsymbol{i}+\boldsymbol{e}_d}-\psi_{\boldsymbol{i}}|,2|\psi_{\boldsymbol{i}}-\psi_{\boldsymbol{i}-\boldsymbol{e}_d}|,\frac{1}{2}|\psi_{\boldsymbol{i}+\boldsymbol{e}_d}-\psi_{\boldsymbol{i}-\boldsymbol{e}_d}|)}{\Delta x_d} & b > 0, \\ 0 & b \leq 0 \end{cases} \tag{4.57}$$

where

$$b = (\psi_{\boldsymbol{i}+\boldsymbol{e}_d} - \psi_{\boldsymbol{i}}) \cdot (\psi_{\boldsymbol{i}} - \psi_{\boldsymbol{i}-\boldsymbol{e}_d}). \tag{4.58}$$

The fourth order accurate slope calculation is as in [31].

4.3.5 Upwinding

Our Riemann solution in (4.39) and (4.45) is simple upwinding,

$$R(\psi^L_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}, \psi^H_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}, u_{d,\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}, d) = \begin{cases} \psi^L_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} & \text{if } u_{d,\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} > 0, \\ \psi^H_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} & \text{if } u_{d,\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} < 0, \\ \frac{1}{2}(\psi^L_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} + \psi^H_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}) & \text{if } u_{d,\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = 0. \end{cases} \tag{4.59}$$

Chapter 5

The Incompressible Navier-Stokes Equations with AMR

In this Section we describe an extension of the single-grid algorithm given in Chapter 4 to the case of block-structured adaptive mesh refinement (AMR). Our approach will be to express the AMR discretizations in terms of the corresponding uniform grid discretizations at each level. An appropriate interpolation operator provides ghost cell values for points in the stencil extending outside of the grids at that level. We will also define a conservative discretization of the needed operators on multilevel data.

For this research we have imposed the following constraint: the embedded boundary interface cannot cross the coarse-fine interface. This additional constraint is for ease of implementation and will be relaxed in future research, similar to [31]. A further constraint imposed for this research is that all AMR levels are advanced with the same time step, i.e. we are not sub-cycling the levels as is done in [1, 72]. In the AMR research community,

methods that do not subcycle AMR levels are termed composite methods, and as such, "mass" conservation is easily maintained with a flux based approach.

5.1 Introduction to Block-Structured AMR

In adaptive methods, one adjusts the computational effort locally to maintain a uniform level of accuracy throughout the problem domain. Refined regions are organized into rectangular patches, as in Figure 5.1. Refinement is possible in both space and time (though for this work all levels use the same time step). AMR allows the simulation of a range of spatial and temporal scales. Capturing these ranges is critical to accurately modeling multiscale transport complexities such as boundaries, fronts, and mixing zones that exist in natural environments. We maintain accuracy and strict conservation with embedded boundaries and AMR. Three basic requirements are necessary to maintain conservation and accuracy with AMR: 1) match fluxes (see Figure 5.2) conservatively at coarse fine interfaces (this leads to a refluxing step for the coarse levels); 2) use interpolation to provide ghost cell values for points in the stencil extending outside of the grids at that level (see Figure 5.3); 3) conservatively coarsen and refine data when regridding.

5.2 EB AMR Notation

We define a coarsening operator by $\mathcal{C}_r : \mathbb{Z}^{\mathbf{D}} \rightarrow \mathbb{Z}^{\mathbf{D}}$,

$$\mathcal{C}_r(\boldsymbol{i}) = (\lfloor \frac{i_0}{r} \rfloor, ..., \lfloor \frac{i_{d-1}}{r} \rfloor) \tag{5.1}$$

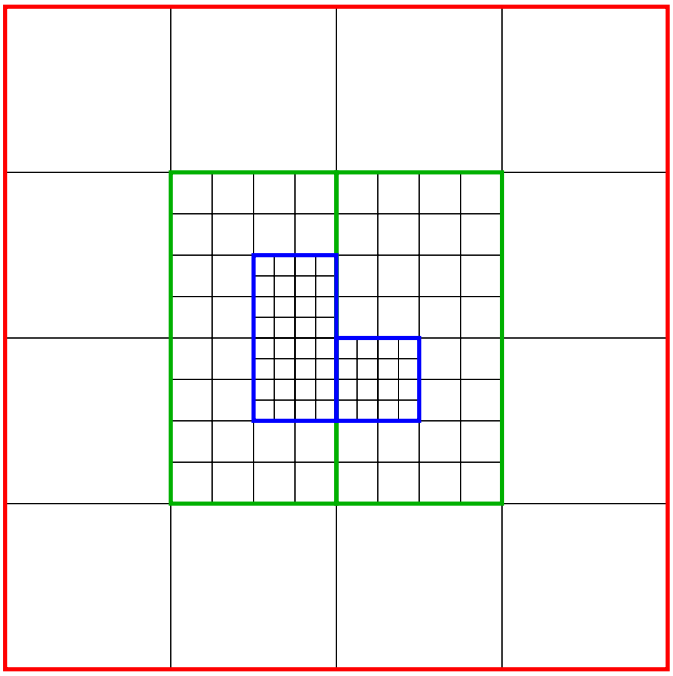

Figure 5.1: Block structured adaptive mesh refinement

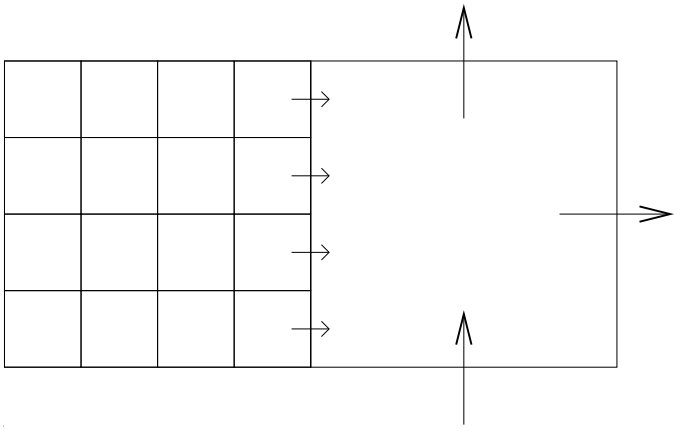

Figure 5.2: Flux matching at coarse-fine interfaces

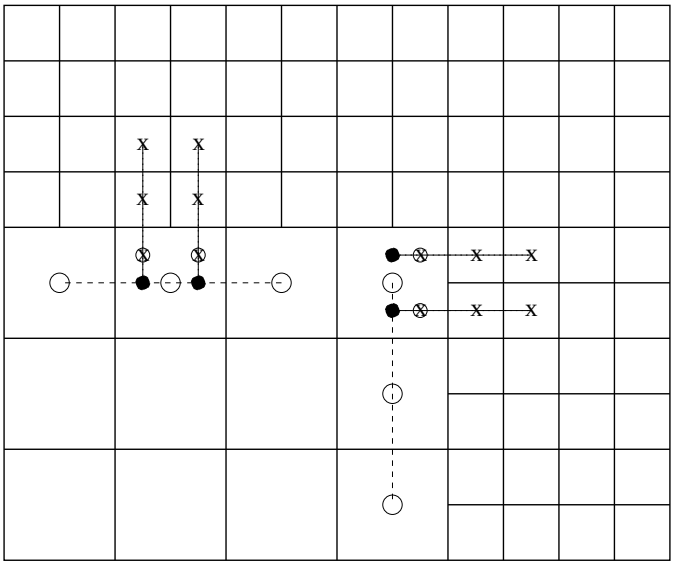

Figure 5.3: Quadratic coarse fine interpolation used for elliptic and parabolic solves

where r is a positive integer. These operators acting on subsets of $\mathbb{Z}^{\mathbf{D}}$ can be extended in a natural way to the face-centered sets: $\mathcal{C}_r(\Gamma^{\boldsymbol{e}^d}) \equiv (\mathcal{C}_r(\Gamma))^{\boldsymbol{e}^d}$. We use a finite-volume discretization of space to represent a nested hierarchy of grids that discretize the same continuous spatial domain. We assume that our problem domain can be discretized by a nested hierarchy of grids $\Gamma^0 ... \Gamma^{lmax}$, with $\Gamma^{l+1} = \mathcal{C}^{-1}_{n^l_{ref}}(\Gamma^l)$. and that the mesh spacings h^l associated with Γ^l satisfy $\frac{h^l}{h^{l+1}} = n^l_{ref}$. The integer n^l_{ref} is the *refinement ratio* between level l and $l+1$. These conditions imply that the underlying continuous spatial domains defined by the control volumes are all identical. In this paper we will further assume n^l_{ref} is even and no less than 2. For any set $\Upsilon \subseteq \Gamma^l$, we define $\mathcal{G}(\Upsilon, r)$, $r > 0$, to the set of all points within a $|\cdot|$-distance r of Υ that are still contained in Γ^l

$$\mathcal{G}(\Upsilon, r) = \Gamma^l \cap \bigcup_{|\boldsymbol{i}| \leq r} \Upsilon + \boldsymbol{i} \tag{5.2}$$

where $|\boldsymbol{i}| = \max_{d=0\ldots\mathbf{D}-1}(|i_d|)$, We can extend the definition to the case $r < 0$

$$\mathcal{G}(\Upsilon, r) = \Gamma^l - \mathcal{G}(\Gamma^l - \Upsilon, -r) \tag{5.3}$$

Thus $\mathcal{G}(\Upsilon, r)$ consists of all of the points in Υ that are within a distance $-r$ from points in the complement of Υ in Γ^l. In the case that there are periodic boundary conditions in one or more of the coordinate directions, we think of the various sets appearing here and in what follows as consisting of the set combined with all of its periodic images for the purpose of defining set operations and computing ghost cell values. For example, $\mathcal{G}(\Upsilon, r)$ is obtained by growing the union of Υ with its periodic images, and performing the intersections and differences with the union of Γ^l with its periodic images.

We make two assumptions about the nesting of grids at successive levels. We require the control volume corresponding to a cell in Ω^{l-1} is either completely contained in the control volumes defined by Ω^l or its intersection has zero volume. We also assume that there is at least n^l_{ref} level l cells separating level $l+1$ cells from level $l-1$ cells: $\mathcal{G}(\mathcal{C}_{n^l_{nref}}(\Omega^{l+1}), n^l_{ref}) \subseteq \Omega^l$. We will refer to grid hierarchies that meet these two conditions as being *properly nested*.

From a formal numerical analysis standpoint, a solution on an AMR hierarchy $\{\Omega^l\}_{l=0}^{l_{max}}$ approximates the exact solution to the PDE only on those cells that are not covered by a grid at a finer level. We define the valid region of Ω^l as,

$$\Omega^l_{valid} = \Omega^l - \mathcal{C}_{n^l_{ref}}(\Omega^{l+1}). \tag{5.4}$$

A composite array ψ^{comp} is a collection of discrete values defined on the valid regions at each of the levels of refinement.

$$\psi^{comp} = \{\psi^{l,valid}\}_{l=0}^{l_{max}}\ ,\ \psi^{l,valid} : \Omega^l_{valid} \rightarrow \mathbb{R}^m \tag{5.5}$$

We can also define valid regions and composite arrays for face-centered variables. $\Omega^{l,\boldsymbol{e}^d}_{valid} = \Omega^{l,\boldsymbol{e}^d} - \mathcal{C}_{n^l_{ref}}(\Omega^{l+1,\boldsymbol{e}^d})$. Thus, $\Omega^{l,\boldsymbol{e}^d}_{valid}$ consists of d-faces that are not covered by the d-faces at the next finer level. A composite vector field $\vec{F}^{comp} = \{\vec{F}^{l,valid}\}^{l_{max}}_{l=0}$ is defined as follows.

$$\vec{F}^{l,valid} = (F_0^{l,valid} \dots F_{\mathbf{D}-1}^{l,valid}) \ , \ F_d^{l,valid} : \Omega^{l,\boldsymbol{e}^d}_{valid} \rightarrow \mathbb{R} \tag{5.6}$$

Thus a composite vector field has values at level l on all of the faces not covered by faces at the next finer level.

5.3 Hyperbolic Equations

As in Section 3.4, we use the explicit hyperbolic methodology developed in [31, 83], with the same minor deviations for incompressible flow outlined in Section (4.3). Specifically, we use the single level hyperbolic methodology with flux matching at the coarse-fine interface, and conservative linear interpolation from coarse levels to fine ghost cells.

5.4 Elliptic and Parabolic Equations

We extend our single-grid solution methodology [92] for solving (3.8), (3.20), and (3.25) by using essentially the same second-order accurate (non-EB) AMR elliptic algorithm as is outlined in [71, 72]. The algorithm pseudo-code is in Figures 5.4-5.6. Note that when we use our line-solver as a multigrid smoother, our additional requirement is that our disjoint box layout have one box per (connected) line per level (with as many grids in the horizontal as needed), and we impose homogeneous boundary conditions at the coarse-fine interface.

$R_{initial} := \rho - L(\phi_{initial})$

while $(||R|| > \epsilon||R_{initial}||)$

VCycleMG(l^{max})

$R := \rho - L(\phi)$

end while

Figure 5.4: Pseudo-code description of the locally-refined multigrid algorithm.

The smoothing operator $mgRelax(\phi^f, R^f, r)$ performs a multigrid V-cycle iteration on ϕ^f for the operator L^{nf}, assuming the coarse-grid values required for the boundary conditions are identically zero. Within $mgRelax()$ our $LevelSmoother()$ is either colored Gauss-Siedel or (for anisotropic mesh spacing) the line solver from Figure A.1, which we apply twice per smooth (in Figure 5.6 we set NumSmoothDown = NumSmoothUp = 4).

5.5 The Incompressible Navier-Stokes Equations

We extend our single-grid solution methodology (see Chapter 4) for solving the incompressible Navier-Stokes equations, by extending the remaining operators (those not outlined in Sections 5.3 and 5.4) in a conservative and second-order accurate manner. The AMR INS pseudo-code is nearly identical to the single-grid algorithm (see Section 4.1) with the exception of regridding and coarse-fine boundary conditions. Specifically, we match fluxes at coarse-fine interfaces (e.g. in the MAC-projection), and use conservative coarsening

Procedure VCycleMG(level l):

if $(l = l^{max})$ then $R^l := \rho^l - L^{nf}(\phi^l, \phi^{l-1})$

if $(l > 0)$ then

$\phi^{l,save} := \phi^l$ on Ω^l

$e^l := 0$ on Ω^l

mgRelax$(e^l, R^l, n_{ref}^{l-1})$

$\phi^l := \phi^l + e^l$

$e^{l-1} := 0$ on Ω^{l-1}

$R^{l-1} :=$ Average$(R^l - L^{nf}(e^l, e^{l-1}))$ on $\mathcal{C}_{n_{ref}^{l-1}}(\Omega^l)$

$R^{l-1} := \rho^{l-1} - L^{comp,l-1}(\phi)$ on $\Omega^{l-1} - \mathcal{C}_{n_{ref}^{l-1}}(\Omega^l)$

VCycleMG$(l-1)$

$e^l := e^l + I_{pwc}(e^{l-1})$

$R^l := R^l - L^{nf,l}(e^l, e^{l-1})$

$\delta e^l := 0$ on Ω^l

mgRelax$(\delta e^l, R^l, n_{ref}^{l-1})$

$e^l := e^l + \delta e^l$

$\phi^l := \phi^{l,save} + e^l$

else

solve $L^{nf}(e^0) = R^0$ on Ω^0.

$\phi^0 := \phi^0 + e^0$

end if

Figure 5.5: Pseudo-code description of the AMR multigrid v-cycle algorithm.

```
procedure mgRelax(φ^f, R^f, r)
{
    for i = 1, ..., NumSmoothDown
        LevelSmoother(φ^f, R^f)
    end for
    if (r > 2) then
        δ^c := 0
        R^c := Average(R^f − L^nf(φ^f, φ^c ≡ 0))
        mgRelax(δ^c, R^c, r/2)
        φ^f := φ^f + I_pwc(δ^c)
        for i = 1, ..., NumSmoothUp
            LevelSmoother(φ^f, R^f)
        end for
    end if
}
```

Figure 5.6: Recursive relaxation procedure.

and refinement when regridding. Recall that for this research we do not permit the coarse-fine interface to cross the embedded boundary, and we do not subcycle the AMR levels (to maintain consistent CFL numbers on each level) as is done in [1, 72].

A pseudo-code description of our composite EB AMR INS algorithm is presented in figure 5.7. For the PiecewiseLinearFillPatch() function we do linear interpolation based on coarse data to fine ghost cells (as in [1, 72]). For the ExtrapToFacesAtHalfTime() function we use our single level infrastructure from Section 4.3. Our discrete composite (comp) operator $\mathbf{P}^{mac,comp}$ is the multilevel analog to the single level $\mathbf{P}^{mac}$ operator. $\mathbf{P}^{mac,comp}$ maintains conservation at coarse-fine interfaces by setting the coarse flux equal to the average of the fine fluxes (as in figure 5.2). The Hyperbolic() function computes the advective terms by level, as in Section 4.3. Our CompositeParabolic() function is as in Section 3.3.3 except we use coarse-fine interpolation and refluxing methodology to couple the levels (see Figures 5.2 and 5.3). The $\mathbf{P}^{cc,comp}$ is related to the $\mathbf{P}^{mac,comp}$ in the same way as for the single level solves (where we interpolate data back and forth between cell- and face-centers). In our Regrid() step we tag control-volumes (e.g. based on vorticity magnitude and density gradients), cluster the tags into disjoint blocks (using [17]), and copy or conservatively linearly-interpolate the old data to the new data-structures (as in [1, 72]).

Initialize($\vec{u}^0, \rho^0, \nabla p^{-\frac{1}{2}}$)

for $n = 0, \ldots,$ NumTimeSteps

{

PiecewiseLinearFillPatch(ρ^n)

PiecewiseLinearFillPatch($\vec{u}^n$)

for $l = 0, \ldots, l^{max}$

{

$\vec{\tilde{u}}^{n+\frac{1}{2},l} = \text{ExtrapToFacesAtHalfTime}(\vec{u}^{n,l}, [\frac{\nabla p}{\rho}]^{n-\frac{1}{2},l})$

}

$\vec{u}^{ADV} = \mathbf{P}^{mac,comp}(\vec{\tilde{u}}^{n+\frac{1}{2}})$

for $l = 0, \ldots, l^{max}$

{

$\left[A[\vec{u}]^{n+\frac{1}{2},l}, A[\rho]^{n+\frac{1}{2},l}\right] = \text{Hyperbolic}(\vec{u}^{n,l}, \vec{u}^{ADV,l}, \rho^{n,l}, [\frac{\nabla p}{\rho}]^{n-\frac{1}{2},l})$

}

$\rho^{n+1} = \rho^n - \Delta t A[\rho]^{n+\frac{1}{2}}$

$\vec{u}^* = \text{CompositeParabolic}(\vec{u}^n, A[\vec{u}]^{n+\frac{1}{2}}, [\frac{\nabla p}{\rho}]^{n-\frac{1}{2}})$

$\left[\vec{u}^{n+1}, [\frac{\nabla p}{\rho}]^{n+\frac{1}{2}}\right] = \mathbf{P}^{cc,comp}(\vec{u}^*, \rho^{n+\frac{1}{2}}, [\frac{\nabla p}{\rho}]^{n-\frac{1}{2}})$

Regrid()

}

Figure 5.7: Pseudo-code description of the adaptive incompressible Navier-Stokes algorithm.

Chapter 6

Results

This chapter demonstrates the accuracy of the method for solving environmental flows. To test accuracy we have selected a series of test problems. The test problems build in difficulty starting with a sphere driven cavity, then a trapped vortex ring, followed by the classic flow past a sphere problem. Subsequently we test the method by simulating complex variable density flows, and finally we test the method on wind driven circulations in Lake Tahoe, USA.

As mentioned in Section 1.3.4, this work was implemented within, and extended the Chombo [32] adaptive mesh refinement framework. Chombo provides the necessary data structures to implement the highly complex adaptive algorithms required for solving the incompressible Navier-Stokes equations in irregular domains. Chombo builds and executes on a range of computational platforms, from laptops to parallel supercomputers. The calculations presented in this Chapter were computed in parallel using SPMD style programming, with multiblocked levels. In this paradigm each decomposition block is assigned

to a parallel processor, communicating ghost cell information as needed via MPI.

6.1 Accuracy Measures

To test the convergence properties of the method, we conduct convergence studies. Since all of our test problems do not have known solutions, we conduct convergence studies using at least 3 sets of grids (coarse, medium, and fine), and compute errors between grids (coarse compared to medium, medium compared to fine). We then can compute norms of these errors, and subsequently the convergence rate of the method for the particular test problem and norm (as in [9, 1, 31]).

We compute volume weighted p-norms as follows:

$$L_p \equiv \|e\|_p = \frac{1}{|\Omega|}\Big(\sum_l \sum_{\boldsymbol{i}\in\Omega^l_{valid}} |e_{\boldsymbol{i}}|^p |V^l_{\boldsymbol{i}}|\Big)^{1/p}, \tag{6.1}$$

where $|\Omega|$ is the volume of the flow domain, and $|V^l_{\boldsymbol{i}}|$ are the individual control volume magnitudes for level l, see (3.1). When comparing fine (e^f) and coarse (e^c) errors, we compute convergence rates as follows:

$$r_p = \frac{\log(\frac{\|e^f\|_p}{\|e^c\|_p})}{\log(2)} \tag{6.2}$$

The expected solution error convergence rates for this method are $r_1 = 2$, $r_2 = 1.5$ and $r_\infty = 1$, as in [31].

6.2 Sphere Driven Cavity

To test the convergence properties of our constant density incompressible Navier-Stokes algorithm (using DWPM1 from Section 4.1.4), we simulate the 3D evolution of the

flow field within a rigidly rotating sphere, as in Figure 6.1. The sphere has a radius of 0.5 $[m]$, and is centered at the origin. The rigid rotation of the sphere is around the z-axis, and is started smoothly with a cubic polynomial as follows

$$\vec{u}^{\text{Sphere}} = [-y, x, 0] f(t), \tag{6.3}$$

$$f(t) = \begin{cases} r\big(-2(\frac{t}{T})^3 + 3(\frac{t}{T})^2\big) & \text{if } t < T, \\ r & \text{if } t \geq T. \end{cases} \tag{6.4}$$

We set $r = 0.1$ and $T = 100$, and our kinematic viscosity is $\nu = 1$, giving an approximately unit Reynolds number for $t = 6$, the stopping time for this test. We impose a no-slip boundary condition on the sphere. This no-slip boundary condition combined with the rigid rotation of the sphere, slowly drives the interior fluid into motion, as in Figure 6.1. The long term state for this flow is rigid rotation of the entire fluid, with constant vorticity parallel to the axis of rotation. In order to test the temporal accuracy of the method, we focus our attention on early times ($t = [0, 6]$). We computed 4, 8, and 16 time steps for the coarse, medium, and fine runs over this timeframe, using 64 parallel processors.

Convergence results from this calculation are presented in Tables 6.1-6.3. The results are as we would expect ($r_\infty = 1$, $r_2 = 1.5$, and $r_1 = 2$) from such a smooth problem. The results are symmetric as is apparent by comparing the U-Velocity and V-Velocity errors in Table 6.2.

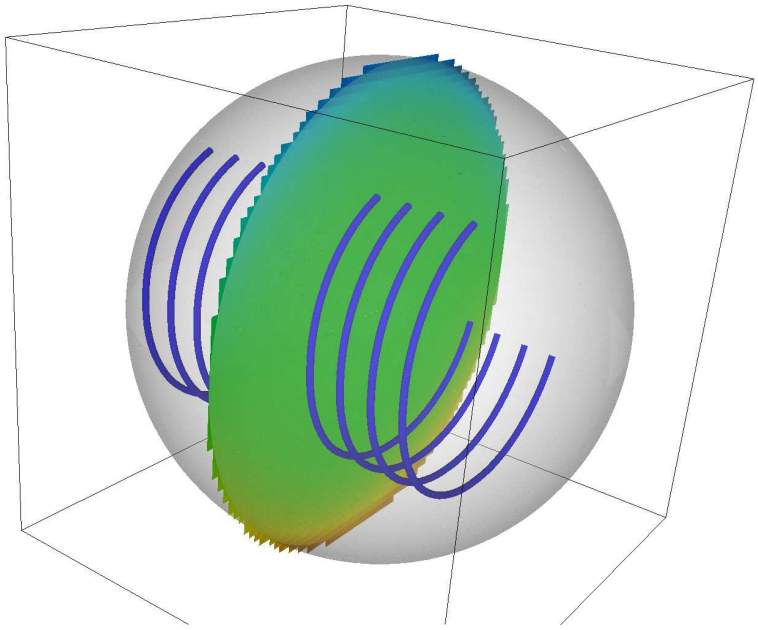

Figure 6.1: Sphere driven cavity. The slice is colored by u-velocity, and streamtubes aid in visualizing the flow.

Table 6.1: Solution error convergence rates using L_∞ norm: $\Delta x_c = \frac{1}{64} = 2\Delta x_m = 4\Delta x_f$; $\Delta x_c = (1.5625e-02, 1.5625e-02, 1.5625e-02) = 2\Delta x_m = 4\Delta x_f$. Isotropic mesh spacing. A well resolved $3D$ viscous calculation.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	5.2068e-05	2.5166e-05	**1.0489e+00**
V-Velocity	5.2068e-05	2.5166e-05	**1.0489e+00**
W-Velocity	3.0969e-06	7.5950e-07	**2.0277e+00**
Pressure	2.3083e-04	6.3841e-05	**1.8543e+00**
Scalar-0	4.0330e-03	2.0201e-03	**9.9740e-01**
Scalar-1	4.0330e-03	2.0201e-03	**9.9740e-01**
Scalar-2	3.9233e-03	1.9653e-03	**9.9734e-01**
p_x/ρ	9.6348e-06	2.2130e-06	**2.1223e+00**
p_y/ρ	9.6348e-06	2.2130e-06	**2.1223e+00**
p_z/ρ	6.8615e-06	1.5156e-06	**2.1786e+00**
X-Vorticity	1.6148e-03	9.3316e-04	**7.9113e-01**
Y-Vorticity	1.6148e-03	9.3316e-04	**7.9113e-01**
Z-Vorticity	2.7636e-03	1.6378e-03	**7.5479e-01**

Table 6.2: Solution error convergence rates using L_1 norm: $\Delta x_c = \frac{1}{64} = 2\Delta x_m = 4\Delta x_f$; $\Delta x_c = (1.5625e-02, 1.5625e-02, 1.5625e-02) = 2\Delta x_m = 4\Delta x_f$. Isotropic mesh spacing. A well resolved $3D$ viscous calculation.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	2.7969e-06	7.9661e-07	**1.8119e+00**
V-Velocity	2.7969e-06	7.9661e-07	**1.8119e+00**
W-Velocity	4.1303e-08	8.7800e-09	**2.2340e+00**
Pressure	1.2735e-05	2.8007e-06	**2.1850e+00**
Scalar-0	5.6574e-05	1.4727e-05	**1.9417e+00**
Scalar-1	5.6574e-05	1.4727e-05	**1.9417e+00**
Scalar-2	4.6372e-05	1.1672e-05	**1.9902e+00**
p_x/ρ	2.5805e-07	2.9925e-08	**3.1082e+00**
p_y/ρ	2.5805e-07	2.9925e-08	**3.1082e+00**
p_z/ρ	8.7864e-08	1.2874e-08	**2.7708e+00**
X-Vorticity	1.8025e-05	5.3106e-06	**1.7631e+00**
Y-Vorticity	1.8025e-05	5.3106e-06	**1.7631e+00**
Z-Vorticity	7.3125e-05	2.1292e-05	**1.7801e+00**

Table 6.3: Solution error convergence rates using L_2 norm: $\Delta x_c = \frac{1}{64} = 2\Delta x_m = 4\Delta x_f$; $\Delta x_c = (1.5625e-02, 1.5625e-02, 1.5625e-02) = 2\Delta x_m = 4\Delta x_f$. Isotropic mesh spacing. A well resolved $3D$ viscous calculation.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	4.6086e-06	1.3910e-06	**1.7282e+00**
V-Velocity	4.6086e-06	1.3910e-06	**1.7282e+00**
W-Velocity	9.4183e-08	2.3178e-08	**2.0227e+00**
Pressure	2.1915e-05	4.6073e-06	**2.2499e+00**
Scalar-0	3.0519e-04	1.0762e-04	**1.5038e+00**
Scalar-1	3.0519e-04	1.0762e-04	**1.5038e+00**
Scalar-2	3.0413e-04	1.0743e-04	**1.5012e+00**
p_x/ρ	5.1444e-07	6.9467e-08	**2.8886e+00**
p_y/ρ	5.1444e-07	6.9467e-08	**2.8886e+00**
p_z/ρ	2.8836e-07	4.6898e-08	**2.6203e+00**
X-Vorticity	4.3166e-05	1.6638e-05	**1.3755e+00**
Y-Vorticity	4.3166e-05	1.6638e-05	**1.3755e+00**
Z-Vorticity	1.2579e-04	4.6048e-05	**1.4498e+00**

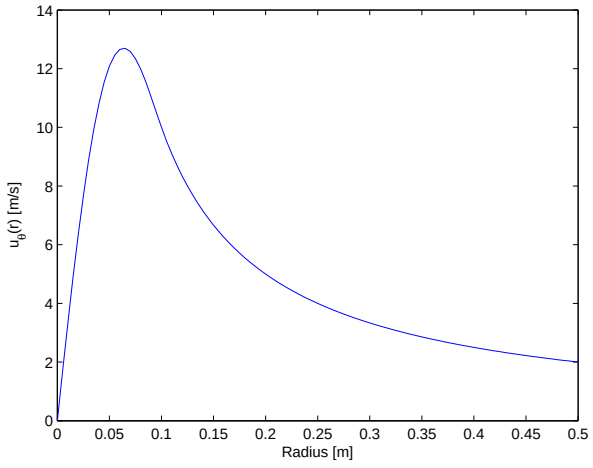

Figure 6.2: Trapped vortex: initial velocity profile.

6.3 Trapped Vortex

To further test the convergence properties of our constant density incompressible Navier-Stokes algorithm (using DWPM1 from Section 4.1.4), we simulate the evolution of a trapped vortex patch in a cylinder (2D), and a vortex ring in a torus (3D).

Our 2D cylinder has a unit diameter with a centered vortex patch specified by the azimuthal velocity component around the vortex center (as in [72]):

$$u_\theta(r) = \begin{cases} \Gamma\left(\frac{8}{3R^5}r^4 - \frac{5}{R^4}r^3 + \frac{10}{3R^2}r\right) & \text{if } r < R, \\ \Gamma\left(\frac{1}{r}\right) & \text{if } r \geq R. \end{cases} \tag{6.5}$$

We choose $R = 0.1$ $[m]$ and $\Gamma = 1$ $[m^2/s]$, and our kinematic viscosity is $\nu = 0.1$ $[m^2/s]$, giving a Reynolds number of $\sim$ 100 based on the maximum velocity and the cylinder diameter. The cross-section velocity profile, given by (6.5), is shown in Figure 6.2.

For the 3D test the torus has a major radius = 0.25 $[m]$, a minor radius = 0.2

$[m]$, with the major-circle normal to the z-axis. We use the same velocity distribution as in (6.5) for the torus cross-sections, except we set $\Gamma = 0.1$ $[m^2/s]$, and project. This makes a symmetric vortex ring trapped inside the torus. See Figure 6.4 for the general layout.

Two-dimensional, viscous single level (16 processor) results are shown in Tables 6.4-6.6, along with a vorticity plot in Figure 6.3. Three-dimensional, viscous single level (64 processor) results are shown in Tables 6.7-6.9. Three-dimensional, inviscid single level (64 processor) results are shown in Tables 6.10-6.12. For the 2D and inviscid results our algorithm indicates the expected accuracy. The 3D viscous results are not yet in the asymptotic regime. Based on these results, the results from Section 6.2, and an examination of the convergence properties of the elliptic operators in Appendix B gives us confidence that our algorithm is fully second order accurate with sufficient mesh refinement. The results are also symmetric as is apparent by comparing the U-Velocity and V-Velocity errors in Tables 6.5 and 6.8.

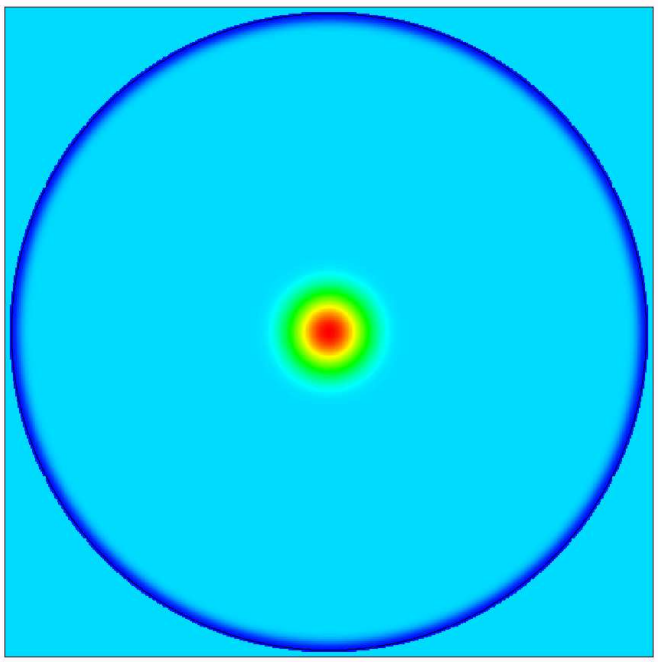

Figure 6.3: Diffused vorticity for the 2D trapped vortex test problem. This is the final time step for the finest grid of Tables 6.4-6.6. Red is counter-clockwise vorticity, blue is clockwise.

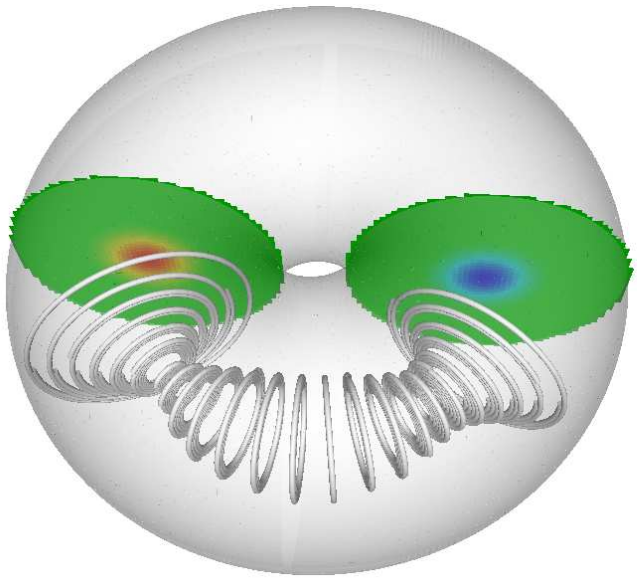

Figure 6.4: Initial condition for the 3D trapped vortex test problem. The torus geometry is shaded grey, slices are colored by vorticity, and streamlines aid in visualizing the flow.

Table 6.4: Solution error convergence rates using L_∞ norm: $\Delta x_c = \frac{1}{256} = 2\Delta x_m = 4\Delta x_f$; $\Delta x_c = (3.9062e-03, 3.9062e-03) = 2\Delta x_m = 4\Delta x_f$. Isotropic mesh spacing. A well resolved $2D$ viscous calculation.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	2.5177e-01	1.0251e-01	**1.2963e+00**
V-Velocity	2.5177e-01	1.0251e-01	**1.2963e+00**
Pressure	1.0325e+03	3.4453e+02	**1.5835e+00**
Scalar-0	1.0713e-03	5.6710e-04	**9.1766e-01**
Scalar-1	1.0713e-03	5.6710e-04	**9.1766e-01**
Scalar-2	1.3617e-03	6.9494e-04	**9.7040e-01**
p_x/ρ	3.1763e+02	4.3731e+01	**2.8606e+00**
p_y/ρ	3.1763e+02	4.3731e+01	**2.8606e+00**
Z-Vorticity	4.4310e+01	8.7482e+00	**2.3406e+00**

Table 6.5: Solution error convergence rates using L_1 norm: $\Delta x_c = \frac{1}{256} = 2\Delta x_m = 4\Delta x_f$; $\Delta x_c = (3.9062e-03, 3.9062e-03) = 2\Delta x_m = 4\Delta x_f$. Isotropic mesh spacing. A well resolved $2D$ viscous calculation.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	4.0903e-03	1.1048e-03	**1.8885e+00**
V-Velocity	4.0903e-03	1.1048e-03	**1.8885e+00**
Pressure	7.8439e+01	2.3260e+01	**1.7537e+00**
Scalar-0	4.6003e-06	1.4196e-06	**1.6963e+00**
Scalar-1	4.6003e-06	1.4196e-06	**1.6963e+00**
Scalar-2	6.2502e-06	1.5698e-06	**1.9933e+00**
p_x/ρ	1.3965e+00	3.5600e-01	**1.9719e+00**
p_y/ρ	1.3965e+00	3.5600e-01	**1.9719e+00**
Z-Vorticity	6.0786e-01	1.4907e-01	**2.0278e+00**

Table 6.6: Solution error convergence rates using L_2 norm: $\Delta x_c = \frac{1}{256} = 2\Delta x_m = 4\Delta x_f$; $\Delta x_c = (3.9062e-03, 3.9062e-03) = 2\Delta x_m = 4\Delta x_f$. Isotropic mesh spacing. A well resolved $2D$ viscous calculation.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	1.4718e-02	3.9476e-03	**1.8985e+00**
V-Velocity	1.4718e-02	3.9476e-03	**1.8985e+00**
Pressure	1.2441e+02	3.8707e+01	**1.6845e+00**
Scalar-0	4.0995e-05	1.4135e-05	**1.5362e+00**
Scalar-1	4.0995e-05	1.4135e-05	**1.5362e+00**
Scalar-2	5.4888e-05	1.9035e-05	**1.5278e+00**
p_x/ρ	4.8248e+00	1.1805e+00	**2.0311e+00**
p_y/ρ	4.8248e+00	1.1805e+00	**2.0311e+00**
Z-Vorticity	2.5748e+00	5.6687e-01	**2.1834e+00**

Table 6.7: Solution error convergence rates using L_∞ norm: $\Delta x_c = \frac{1}{64} = 2\Delta x_m = 4\Delta x_f$; $\Delta x_c = (1.5625e-02, 1.5625e-02, 1.5625e-02) = 2\Delta x_m = 4\Delta x_f$. Isotropic mesh spacing. An under-resolved $3D$ viscous calculation.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	1.0792e-01	1.6430e-01	**-6.0637e-01**
V-Velocity	1.0792e-01	1.6430e-01	**-6.0637e-01**
W-Velocity	2.5021e-01	3.4778e-01	**-4.7506e-01**
Pressure	2.3805e+03	1.9970e+03	**2.5339e-01**
Scalar-0	3.9950e-03	1.9816e-03	**1.0115e+00**
Scalar-1	3.9950e-03	1.9816e-03	**1.0115e+00**
Scalar-2	3.9699e-03	1.9786e-03	**1.0046e+00**
p_x/ρ	7.8310e+01	2.5794e+02	**-1.7197e+00**
p_y/ρ	7.8310e+01	2.5794e+02	**-1.7197e+00**
p_z/ρ	9.8096e+01	2.9420e+02	**-1.5845e+00**
X-Vorticity	3.3221e+01	6.2649e+01	**-9.1522e-01**
Y-Vorticity	3.3221e+01	6.2649e+01	**-9.1522e-01**
Z-Vorticity	8.7073e+00	4.0850e+00	**1.0919e+00**

Table 6.8: Solution error convergence rates using L_1 norm: $\Delta x_c = \frac{1}{64} = 2\Delta x_m = 4\Delta x_f$; $\Delta x_c = (1.5625e-02, 1.5625e-02, 1.5625e-02) = 2\Delta x_m = 4\Delta x_f$. Isotropic mesh spacing. An under-resolved $3D$ viscous calculation.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	5.1309e-03	4.4990e-03	**1.8960e-01**
V-Velocity	5.1309e-03	4.4990e-03	**1.8960e-01**
W-Velocity	9.6873e-03	8.5772e-03	**1.7560e-01**
Pressure	3.8672e+02	3.2154e+01	**3.5882e+00**
Scalar-0	6.3171e-05	1.6931e-05	**1.8996e+00**
Scalar-1	6.3171e-05	1.6931e-05	**1.8996e+00**
Scalar-2	1.0389e-04	2.9203e-05	**1.8308e+00**
p_x/ρ	1.9535e+00	1.0531e+00	**8.9146e-01**
p_y/ρ	1.9535e+00	1.0531e+00	**8.9146e-01**
p_z/ρ	5.5300e+00	1.5353e+00	**1.8487e+00**
X-Vorticity	1.0141e+00	6.0988e-01	**7.3364e-01**
Y-Vorticity	1.0141e+00	6.0988e-01	**7.3364e-01**
Z-Vorticity	5.4883e-02	2.5142e-02	**1.1263e+00**

Table 6.9: Solution error convergence rates using L_2 norm: $\Delta x_c = \frac{1}{64} = 2\Delta x_m = 4\Delta x_f$; $\Delta x_c = (1.5625e-02, 1.5625e-02, 1.5625e-02) = 2\Delta x_m = 4\Delta x_f$. Isotropic mesh spacing. An under-resolved $3D$ viscous calculation.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	1.3661e-02	1.2801e-02	**9.3854e-02**
V-Velocity	1.3661e-02	1.2801e-02	**9.3854e-02**
W-Velocity	1.9847e-02	1.7744e-02	**1.6159e-01**
Pressure	4.7459e+02	5.1100e+01	**3.2153e+00**
Scalar-0	3.4360e-04	1.2136e-04	**1.5014e+00**
Scalar-1	3.4360e-04	1.2136e-04	**1.5014e+00**
Scalar-2	4.8007e-04	1.8192e-04	**1.3999e+00**
p_x/ρ	4.5513e+00	3.5920e+00	**3.4147e-01**
p_y/ρ	4.5513e+00	3.5920e+00	**3.4147e-01**
p_z/ρ	8.1204e+00	5.4910e+00	**5.6449e-01**
X-Vorticity	2.8051e+00	1.6048e+00	**8.0566e-01**
Y-Vorticity	2.8051e+00	1.6048e+00	**8.0566e-01**
Z-Vorticity	2.3705e-01	1.1168e-01	**1.0859e+00**

Table 6.10: Solution error convergence rates using L_∞ norm: $\Delta x_c = \frac{1}{64} = 2\Delta x_m = 4\Delta x_f$; $\Delta x_c = (1.5625e-02, 1.5625e-02, 1.5625e-02) = 2\Delta x_m = 4\Delta x_f$. Isotropic mesh spacing. A $3D$ inviscid calculation.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	2.0324e-02	9.3188e-03	**1.1250e+00**
V-Velocity	2.0324e-02	9.3188e-03	**1.1250e+00**
W-Velocity	3.5406e-02	1.7446e-02	**1.0211e+00**
Pressure	9.6541e-02	6.3686e-02	**6.0016e-01**
Scalar-0	3.9152e-03	1.9584e-03	**9.9941e-01**
Scalar-1	3.9152e-03	1.9584e-03	**9.9941e-01**
Scalar-2	3.9262e-03	1.9632e-03	**9.9990e-01**
p_x/ρ	4.1635e+00	6.7358e+00	**-6.9404e-01**
p_y/ρ	4.1635e+00	6.7358e+00	**-6.9404e-01**
p_z/ρ	2.9866e+00	4.0626e+00	**-4.4387e-01**
X-Vorticity	1.1350e+00	9.4669e-01	**2.6171e-01**
Y-Vorticity	1.1350e+00	9.4669e-01	**2.6171e-01**
Z-Vorticity	3.2670e-01	8.9022e-01	**-1.4462e+00**

Table 6.11: Solution error convergence rates using L_1 norm: $\Delta x_c = \frac{1}{64} = 2\Delta x_m = 4\Delta x_f$; $\Delta x_c = (1.5625e-02, 1.5625e-02, 1.5625e-02) = 2\Delta x_m = 4\Delta x_f$. Isotropic mesh spacing. A $3D$ inviscid calculation.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	9.3917e-04	2.4954e-04	**1.9121e+00**
V-Velocity	9.3917e-04	2.4954e-04	**1.9121e+00**
W-Velocity	1.8197e-03	4.8655e-04	**1.9031e+00**
Pressure	1.9053e-02	2.8225e-03	**2.7549e+00**
Scalar-0	6.2799e-05	1.5643e-05	**2.0052e+00**
Scalar-1	6.2799e-05	1.5643e-05	**2.0052e+00**
Scalar-2	1.0293e-04	2.6434e-05	**1.9611e+00**
p_x/ρ	2.8776e-01	5.0622e-02	**2.5071e+00**
p_y/ρ	2.8776e-01	5.0622e-02	**2.5071e+00**
p_z/ρ	5.5440e-01	8.9734e-02	**2.6272e+00**
X-Vorticity	7.3061e-02	1.9019e-02	**1.9416e+00**
Y-Vorticity	7.3061e-02	1.9019e-02	**1.9416e+00**
Z-Vorticity	7.7648e-03	2.4323e-03	**1.6746e+00**

Table 6.12: Solution error convergence rates using L_2 norm: $\Delta x_c = \frac{1}{64} = 2\Delta x_m = 4\Delta x_f$; $\Delta x_c = (1.5625e-02, 1.5625e-02, 1.5625e-02) = 2\Delta x_m = 4\Delta x_f$. Isotropic mesh spacing. A $3D$ inviscid calculation.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	1.4487e-03	4.3528e-04	**1.7347e+00**
V-Velocity	1.4487e-03	4.3528e-04	**1.7347e+00**
W-Velocity	2.7949e-03	8.0503e-04	**1.7957e+00**
Pressure	2.5498e-02	4.1286e-03	**2.6267e+00**
Scalar-0	3.4337e-04	1.2102e-04	**1.5045e+00**
Scalar-1	3.4337e-04	1.2102e-04	**1.5045e+00**
Scalar-2	4.7969e-04	1.8141e-04	**1.4028e+00**
p_x/ρ	4.4298e-01	8.6189e-02	**2.3617e+00**
p_y/ρ	4.4298e-01	8.6189e-02	**2.3617e+00**
p_z/ρ	7.7711e-01	1.3892e-01	**2.4839e+00**
X-Vorticity	1.4829e-01	3.8058e-02	**1.9622e+00**
Y-Vorticity	1.4829e-01	3.8058e-02	**1.9622e+00**
Z-Vorticity	1.5044e-02	4.6584e-03	**1.6913e+00**

6.4 Flow Past a Cylinder/Sphere

Now we test our constant density incompressible Navier-Stokes method with AMR and anisotropic mesh spacing (using our adaptive version of DWPM1 from Section 4.1.4). We simulate flow past a sphere, a classic problem from fluid mechanics. The Reynolds number is defined as $\mathbf{R} = \frac{UD}{\nu} = 10$ for this test problem. Where U is the far-field flow velocity, D is the sphere diameter, and ν is the kinematic viscosity.

Two-dimensional, unit aspect ratio, 8 processor, AMR results are shown in Tables 6.13-6.15. Two-dimensional, 4:1 aspect ratio, AMR results are shown in Tables 6.16-6.18. Note that these results use our higher-order accurate Dirichlet EB stencil for the viscous operators. We have observed a connection between the poor behavior of our pressure gradient errors and the higher-order stencil. The results of Sections 6.3 and 6.5.2 test the algorithm with the lower-order Dirichlet EB stencil and confirm this behavior. From the results in these Tables our algorithm is indicating the expected accuracy.

Isotropic ($\Delta x_d = \Delta x_{d'}$) 3D results are shown in Tables 6.19-6.21. Anisotropic 4:4:1 ($\Delta x_d \neq \Delta x_{d'}$) 3D results are shown in Tables 6.22-6.24. It is clear that we are not in the asymptotic regime for this 3D problem. These 3D results combined with the 2D results, and the results from Sections 6.2-6.3 give us confidence that our 3D algorithm is giving us the expected accuracy in the asymptotic limit of decreasing mesh spacing.

Table 6.13: Solution error convergence rates using L_∞ norm: $h_f = \frac{1}{64}$ and $h_c = 2h_f$; AMR nref = 4 with 4 levels . Isotropic mesh spacing. A $2D$ viscous calculation.

Variable	Coarse Error	Fine Error	Order
velocity0	1.134043e+01	2.520019e+00	**2.169968e+00**
velocity1	7.718079e+00	1.820990e+00	**2.083519e+00**
pressure	4.116954e+07	8.985082e+06	**2.195974e+00**
scalar1	1.517485e-02	4.716579e-03	**1.685870e+00**
p_x/ρ	1.301402e+07	3.342936e+07	**-1.361049e+00**
p_y/ρ	2.525169e+07	8.556613e+07	**-1.760660e+00**
z-Vorticity	9.128578e+04	1.996422e+04	**2.192974e+00**

Table 6.14: Solution error convergence rates using L_1 norm. $h_f = \frac{1}{64}$ and $h_c = 2h_f$; AMR nref = 4 with 4 levels . Isotropic mesh spacing. A $2D$ viscous calculation.

Variable	Coarse Error	Fine Error	Order
velocity0	1.186156e-02	2.903117e-03	**2.030619e+00**
velocity1	9.707874e-03	2.397566e-03	**2.017585e+00**
pressure	3.990522e+06	6.852523e+05	**2.541871e+00**
scalar1	9.028065e-04	2.401360e-04	**1.910565e+00**
p_x/ρ	1.757296e+04	3.586594e+03	**2.292671e+00**
p_y/ρ	1.545766e+04	2.956564e+03	**2.386328e+00**
z-Vorticity	9.970603e+00	1.790908e+00	**2.476989e+00**

Table 6.15: Solution error convergence rates using L_2 norm. $h_f = \frac{1}{64}$ and $h_c = 2h_f$; AMR nref = 4 with 4 levels . Isotropic mesh spacing. A $2D$ viscous calculation.

Variable	Coarse Error	Fine Error	Order
velocity0	6.242005e-02	1.458440e-02	**2.097584e+00**
velocity1	4.208083e-02	9.503073e-03	**2.146697e+00**
pressure	5.196020e+06	8.734079e+05	**2.572679e+00**
scalar1	2.796191e-03	7.024019e-04	**1.993094e+00**
p_x/ρ	6.594240e+04	4.774662e+04	**4.658079e-01**
p_y/ρ	8.511134e+04	1.016273e+05	**-2.558643e-01**
z-Vorticity	4.179723e+02	6.649610e+01	**2.652066e+00**

Table 6.16: Solution error convergence rates using L_∞ norm: $\Delta x_c = \frac{1}{16} = 2\Delta x_m = 4\Delta x_f$; $\Delta x_c = (3.1250e-02, 3.1250e-02) = 2\Delta x_m = 4\Delta x_f$; a $2D$ viscous calculation with 3 AMR levels having nref = 4. Isotropic mesh spacing.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	6.4016e+00	1.8093e+00	**1.8230e+00**
V-Velocity	4.9507e+00	1.2994e+00	**1.9297e+00**
Pressure	2.3760e+06	3.4213e+05	**2.7959e+00**
Scalar	4.1617e-02	2.3277e-02	**8.3829e-01**
p_x/ρ	9.5118e+04	2.3222e+05	**-1.2877e+00**
p_y/ρ	1.8352e+05	4.6472e+05	**-1.3404e+00**
Z-Vorticity	4.5703e+03	9.5683e+02	**2.2559e+00**

Table 6.17: Solution error convergence rates using L_1 norm: $\Delta x_c = \frac{1}{16} = 2\Delta x_m = 4\Delta x_f$; $\Delta x_c = (3.1250e-02, 3.1250e-02) = 2\Delta x_m = 4\Delta x_f$; a $2D$ viscous calculation with 3 AMR levels having nref = 4. Isotropic mesh spacing.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	2.4871e-02	4.4748e-03	**2.4746e+00**
V-Velocity	2.3543e-02	5.1497e-03	**2.1927e+00**
Pressure	3.2395e+05	2.5091e+04	**3.6905e+00**
Scalar	1.8842e-03	4.8485e-04	**1.9583e+00**
p_x/ρ	1.1697e+03	1.5246e+02	**2.9396e+00**
p_y/ρ	8.0536e+02	9.3514e+01	**3.1064e+00**
Z-Vorticity	8.1307e+00	1.4680e+00	**2.4696e+00**

Table 6.18: Solution error convergence rates using L_2 norm: $\Delta x_c = \frac{1}{16} = 2\Delta x_m = 4\Delta x_f$; $\Delta x_c = (3.1250e-02, 3.1250e-02) = 2\Delta x_m = 4\Delta x_f$; a $2D$ viscous calculation with 3 AMR levels having nref = 4. Isotropic mesh spacing.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	1.2722e-01	2.8571e-02	**2.1547e+00**
V-Velocity	8.7780e-02	1.8550e-02	**2.2425e+00**
Pressure	4.1739e+05	3.2361e+04	**3.6891e+00**
Scalar	6.2881e-03	1.9745e-03	**1.6711e+00**
p_x/ρ	1.9105e+03	5.2372e+02	**1.8671e+00**
p_y/ρ	1.9859e+03	6.7675e+02	**1.5531e+00**
Z-Vorticity	9.5604e+01	1.0532e+01	**3.1823e+00**

Table 6.19: Solution error convergence rates using L_∞ norm: $h_c = \frac{1}{32} = 2h_m = 4h_f$; a $3D$ viscous calculation.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	2.4454e+01	7.2745e+00	**1.7492e+00**
V-Velocity	8.3777e+00	1.5826e+01	**-9.1768e-01**
W-Velocity	8.3777e+00	1.5826e+01	**-9.1768e-01**
Pressure	2.9432e+05	4.6078e+05	**-6.4669e-01**
Scalar	1.0824e-01	5.3912e-02	**1.0055e+00**
p_x/ρ	4.7156e+04	1.1739e+05	**-1.3158e+00**
p_y/ρ	8.7238e+04	5.8004e+05	**-2.7331e+00**
p_z/ρ	8.7238e+04	5.8004e+05	**-2.7331e+00**
X-Vorticity	4.6583e+02	1.2122e+03	**-1.3797e+00**
Y-Vorticity	2.8827e+03	1.1753e+03	**1.2944e+00**
Z-Vorticity	2.8827e+03	1.1753e+03	**1.2944e+00**

Table 6.20: Solution error convergence rates using L_1 norm: $h_c = \frac{1}{32} = 2h_m = 4h_f$; a $3D$ viscous calculation.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	1.0689e-01	3.4822e-02	**1.6181e+00**
V-Velocity	4.4420e-02	1.4482e-02	**1.6170e+00**
W-Velocity	4.4420e-02	1.4482e-02	**1.6170e+00**
Pressure	1.7750e+04	1.2839e+04	**4.6723e-01**
Scalar	1.8962e-04	5.2633e-05	**1.8491e+00**
p_x/ρ	9.1412e+01	7.1387e+01	**3.5672e-01**
p_y/ρ	5.4115e+01	6.0387e+01	**-1.5822e-01**
p_z/ρ	5.4115e+01	6.0387e+01	**-1.5822e-01**
X-Vorticity	3.6631e-01	2.6211e-01	**4.8291e-01**
Y-Vorticity	3.5284e+00	1.4344e+00	**1.2986e+00**
Z-Vorticity	3.5284e+00	1.4344e+00	**1.2986e+00**

Table 6.21: Solution error convergence rates using L_2 norm: $h_c = \frac{1}{32} = 2h_m = 4h_f$; a $3D$ viscous calculation.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	3.1787e-01	1.2120e-01	**1.3911e+00**
V-Velocity	1.1361e-01	7.5938e-02	**5.8122e-01**
W-Velocity	1.1361e-01	7.5938e-02	**5.8122e-01**
Pressure	2.2758e+04	1.6582e+04	**4.5675e-01**
Scalar	1.3957e-03	4.0632e-04	**1.7803e+00**
p_x/ρ	1.9389e+02	5.8685e+02	**-1.5978e+00**
p_y/ρ	1.6036e+02	2.2072e+03	**-3.7828e+00**
p_z/ρ	1.6036e+02	2.2072e+03	**-3.7828e+00**
X-Vorticity	4.7074e+00	9.1630e+00	**-9.6089e-01**
Y-Vorticity	2.4620e+01	1.4009e+01	**8.1342e-01**
Z-Vorticity	2.4620e+01	1.4009e+01	**8.1342e-01**

Table 6.22: Solution error convergence rates using L_∞ norm: $\Delta x_c = \frac{1}{16} = 2\Delta x_m = 4\Delta x_f$; $\Delta x_c = (3.1250e-02, 3.1250e-02, 1.5625e-02) = 2\Delta x_m = 4\Delta x_f$; a $3D$ viscous calculation. Aspect ratio: dx/dz = 2.0000e+00 and dy/dz = 2.0000e+00.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	2.1219e+01	1.8371e+01	**2.0796e-01**
V-Velocity	6.7553e+00	6.8038e+00	**-1.0331e-02**
W-Velocity	6.1780e+00	5.4981e+00	**1.6821e-01**
Pressure	2.4518e+05	1.5432e+05	**6.6793e-01**
Scalar	2.2718e-01	8.6465e-02	**1.3936e+00**
p_x/ρ	1.1648e+04	1.4950e+04	**-3.6005e-01**
p_y/ρ	1.1711e+04	1.4997e+04	**-3.5679e-01**
p_z/ρ	1.1278e+04	1.6250e+04	**-5.2697e-01**
X-Vorticity	6.5421e+02	4.8398e+02	**4.3480e-01**
Y-Vorticity	2.2321e+03	1.8535e+03	**2.6817e-01**
Z-Vorticity	1.1514e+03	2.4019e+03	**-1.0608e+00**

Table 6.23: Solution error convergence rates using L_1 norm: $\Delta x_c = \frac{1}{16} = 2\Delta x_m = 4\Delta x_f$; $\Delta x_c = (3.1250e-02, 3.1250e-02, 1.5625e-02) = 2\Delta x_m = 4\Delta x_f$; a $3D$ viscous calculation. Aspect ratio: dx/dz = 2.0000e+00 and dy/dz = 2.0000e+00.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	9.3345e-02	7.8160e-02	**2.5615e-01**
V-Velocity	3.8943e-02	3.2503e-02	**2.6077e-01**
W-Velocity	4.1218e-02	3.2885e-02	**3.2585e-01**
Pressure	2.4353e+04	3.7301e+03	**2.7068e+00**
Scalar	4.4734e-04	1.2647e-04	**1.8226e+00**
p_x/ρ	1.2393e+02	3.8866e+01	**1.6729e+00**
p_y/ρ	7.1204e+01	2.5826e+01	**1.4631e+00**
p_z/ρ	8.0233e+01	3.5804e+01	**1.1641e+00**
X-Vorticity	6.3006e-01	3.7817e-01	**7.3646e-01**
Y-Vorticity	3.8489e+00	2.7873e+00	**4.6559e-01**
Z-Vorticity	4.3630e+00	2.7256e+00	**6.7877e-01**

Table 6.24: Solution error convergence rates using L_2 norm: $\Delta x_c = \frac{1}{16} = 2\Delta x_m = 4\Delta x_f$; $\Delta x_c = (3.1250e-02, 3.1250e-02, 1.5625e-02) = 2\Delta x_m = 4\Delta x_f$; a $3D$ viscous calculation. Aspect ratio: dx/dz = 2.0000e+00 and dy/dz = 2.0000e+00.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	2.7012e-01	2.5453e-01	**8.5775e-02**
V-Velocity	1.1499e-01	1.0781e-01	**9.2965e-02**
W-Velocity	1.2823e-01	7.9341e-02	**6.9254e-01**
Pressure	3.1216e+04	5.4582e+03	**2.5158e+00**
Scalar	2.9092e-03	9.6271e-04	**1.5955e+00**
p_x/ρ	2.4777e+02	1.4697e+02	**7.5346e-01**
p_y/ρ	1.7465e+02	1.2474e+02	**4.8553e-01**
p_z/ρ	2.0983e+02	1.7410e+02	**2.6926e-01**
X-Vorticity	6.2871e+00	4.6007e+00	**4.5056e-01**
Y-Vorticity	2.5584e+01	2.0751e+01	**3.0212e-01**
Z-Vorticity	3.1442e+01	2.0918e+01	**5.8794e-01**

6.5 Internal Wave Generation and Dissipation

Here we test the incompressible Navier-Stokes algorithm with buoyancy forcing. The goal is to test the accuracy of the method in simulating highly non-linear, nonBoussinesq, density driven flows.

6.5.1 Internal Wave Background and Relevant Literature

Oceanic surface tides and lake motions can induce currents over topography which in turn generate internal gravity waves that propagate along density gradients beneath the surface.[1] While the amplitude of oceanic surface waves due to the tides and lake seiches due to wind forcing are typically less than meters, resulting internal wave amplitudes can be larger than 100 meters, and their associated currents can alter surface characteristics enough to make them visible from space. Their ubiquitous nature is documented in the Global Ocean Associates Internal Wave Atlas [58], which is a compilation of satellite images of internal waves taken from a variety of sources.

Like surface waves, internal waves propagate until they dissipate at bathymetric boundaries, and in the oceanic context there is strong evidence in support of the idea that internal wave breaking at boundaries results in mixed fluid which propagates out into the ocean interior [82, 101, 40, 81], effectively preventing the ocean from turning into a "stagnant pool of cold, salty water" [81]. Developing models for mixing induced by internal waves is therefore paramount to accurate predictions of circulation. If improved models for internal wave processes are to be developed, this requires an accurate understanding of

[1] This Sub-Section (6.5.1) is adapted from a proposal prepared with Professor Oliver Fringer of Stanford University.

their physics, which includes their generation mechanisms, how they propagate and interact with each other and bathymetry, and how they ultimately end up dissipating their energy through breaking. The greatest impediment to the numerical simulation of internal waves is the broad range of length scales over which they exist. This is not surprising for most numerical simulations especially when dissipation scales are involved. However, internal waves still present a challenging computational problem simply due to the range of scales which govern their propagation, even if their dissipative scales are not to be captured.

Internal wave generation

Internal wave energy can exist in the form of solitary waves that propagate as first-mode waves along the pycnocline (region of rapidly changing density). In the oceanic context, solitary waves can be generated as a result of the interaction of barotropic tides with steep topography, as described in the pioneering laboratory experiments of Maxworthy [74]. Maxworthy induced an oscillatory flow over a three-dimensional hill and showed that a quasi-steady depression is formed on the lee side of the hill during the ebb phase of the flow, as depicted in Figure 6.5(a). At some point in the flow, the phase speed c of the depression wave exceeds the flow speed u, and the wave begins to propagate upstream, as shown in Figure 6.5(b). This wave continues to propagate upstream upon current reversal and forms a train of rank-ordered solitary waves, as in Figure 6.5(c), that ride on the flood current u.

Hibiya [53, 54] later determined that the generation process is unsteady and results from the superposition of internal wave characteristics, which was verified numerically and experimentally by Matsuura and Hibiya [73]. In these laboratory experiments, they de-

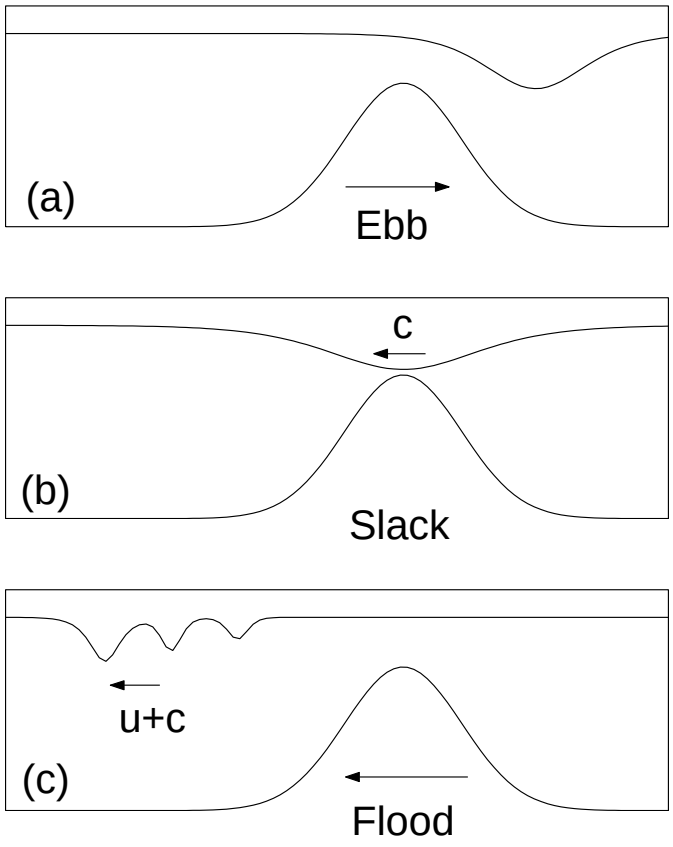

Figure 6.5: Schematic of the internal wave generation mechanism as described by Maxworthy (1979). The variable curved line indicates the approximate pycnocline location.

scribed the generation process as being highly dependent on the Froude number $Fr_m = u/c$, where u is the current speed and c is the internal wave speed over the topography, and $T_d = cT/2b$ is the ratio of the tidal timescale to the internal wave timescale over the topography, where T is the tidal period and b is the topographic half-width. The parameter T_d determines whether waves formed during the flood phase of the tide will interact with those formed during the ebb phase of the tide, since if $T_d >> 1$ waves formed during the ebb phase will be far from the generation site during the flood phase of the tide. The simulations of Matsuura and Hibiya employed $T_d = O(1)$ in order to simulate the generation process on Stellwagen Bank, where $T_d \approx 1.5$ and $Fr_m \approx 1.75$.

First-mode solitary waves are also generated over shelf breaks in a manner similar to the generation mechanism over sills. Lamb [65] employed two-dimensional simulations of the shelf break at Georges Bank and showed that the generation mechanism is due to the formation of a depression wave during supercritical off-bank flow ($Fr_m > 1$) during the ebb tide. This leads to the formation of first- and second-mode internal wave energy that propagates on- and off-shelf. Lamb demonstrated that rotation had a significant effect on the generation process, since a rotational adjustment caused the formation of a second on-bank depression after the first depression.

A classic example of lee-wave generation over a sill occurs in the Sulu Sea, between Malaysia and the Philippines [4]. In this case there is a sill at the southern end of the Sea with a characteristic width of 2 km, and the strong ebb tides there induce currents in excess of 3.4 m s^{-1}. These supercritical currents create a 5 to 10 km depression in the lee of the sill with an amplitude of 50 to 150 m. During the weaker flood tide, this lee wave

propagates northward into the Sulu Sea as a strong bore with a length of 50 to 90 km and an amplitude of 50 to 90 m, with a phase speed of 2.2 m s^{-1}. This bore breaks up into a train of solitary waves that spreads out radially into the sea with crest lengths in excess of 350 km. Based on these length scales, $T_d = 84$, which is much greater than the aforementioned studies of Maxworthy [74], Hibiya [54], and Lamb [65], which is due to the extremely short topographic length scale. This accentuates the need for adaptive mesh refinement to study such internal waves numerically.

Propagation

Far from the generation zone, internal wave energy is predominantly in the form of low modes, since energy in the form of higher modes usually dissipates closer to the internal generation site. For example, low-mode internal wave energy has been observed up to 1000 km away from generation at the Hawaiian Ridge [88].

When the amplitude becomes large with respect to the wavelength, asymptotic methods [96, 67, 43, 3, 24] break down and more complex methods must be used for their simulation. Increased degrees of complexity that allow for mechanisms such as overturning, steep bathymetry, mixing, or viscosity are required in order to accurately represent the physics. This requires the solution of full hydrodynamic equation sets without asymptotic expansions.

Dissipation and breaking

A great deal of our understanding of dissipative processes in breaking internal waves has been achieved with laboratory experiments. Employing continuous stratification,

Cacchione and Wunsch [22] used a first-mode wave propagating towards sloped bathymetry to show that at the critical angle, or the angle at which the bathymetric slope matches the angle of propagation of the internal wave beams, strong shearing motion leads to instability and results in the formation of a series of vortices, which upon overturning mix fluid locally. The mixed fluid then propagates into the interior of the domain along isopycnals, a process called lateral homogenization. Other researchers have also performed similar experiments using continuous stratification [57, 38] and found that the breaking process leads to a highly nonlinear upslope-propagating bolus which leads to a bulk of the dissipation and mixing. The upslope-propagating bolus was also found in the interfacial solitary wave experiments of Helfrich [52], who studied the interaction of solitary wave packets with a uniform slope and showed that each solitary wave generates an upslope-propagating bolus. Michallet and Ivey [77] also observed the upslope-propagating bolus when a single solitary wave of depression interacts with a uniform slope.

In the field, clear signatures of the interaction of highly nonlinear waves with a shelf break and the subsequent dissipation and mixing have rarely been observed, since they exist in the later, dissipative stages of the internal wave life cycle when turbulence and mixing weaken detectable internal wave signatures. Recently, however, some striking measurements of a series of bottom-trapped solitary waves of elevation have been obtained on the Oregon Shelf by Klymak and Moum [63] and near the Massachusetts coast by Scotti and Pineda [93]. These measurements depict trains of highly nonlinear solitary waves of elevation that propagate along the seafloor. With high velocities near the bed, Klymak and Moum propose that bottom stress is likely to play an important role in wave energy

dissipation, unlike near-surface solitary waves, where shear instability at the interface is the primary source of turbulence and dissipation [80], or progressive interfacial waves, where shear at the interface also dominates the dissipation [44, 103]. As examples of other bottom-trapped internal waves of elevation, Hosegood et al. [56] have measured bottom-trapped solibores in the Faeroe-Shetland Channel, while Carter et al. [23] have observed highly nonlinear solitary waves of elevation in Monterey Bay which propagate towards the shelf break. While the generation mechanism for these waves is not yet clear, dissipation and mixing levels within them is orders of magnitude higher than background levels.

6.5.2 Stratified Flow Past a Sill

Here we test our ability to generate internal waves. Using our 2D version of DWPM2 from Section 4.1.4, we simulate a field-scale, idealized, internal wave generating sill (as in Figure 6.5). This type of problem has been studied by [65, 41, 36], and is common in oceanographic (and even atmospheric) settings where currents force a stratified profile past topographic features.

The 2D domain is 256 [m] deep by 4096 [m] long, with a Gaussian sill centered 1024 meters from the left side. The Gaussian is described by, $H \exp^{\frac{-x^2}{2\sigma^2}}$, with $\sigma = 10^5$ $[m^2]$, and the sill height $H = 196$ [m]. This makes the shallowest spot above the sill 60 [m] deep. The domain is forced by a constant inflowing current from the left face at 0.2 $[m/s]$ (which is in the range of tidal currents). The stably stratified initial and inflowing density profile is given by $\rho = 1001 - \exp^{0.0673z}$ $[kg/m^3]$, where $z = 0$ is at the top of the domain. The initial state is shown at the top of Figure 6.7, while significant features of the developed flow are shown in Figure 6.6.

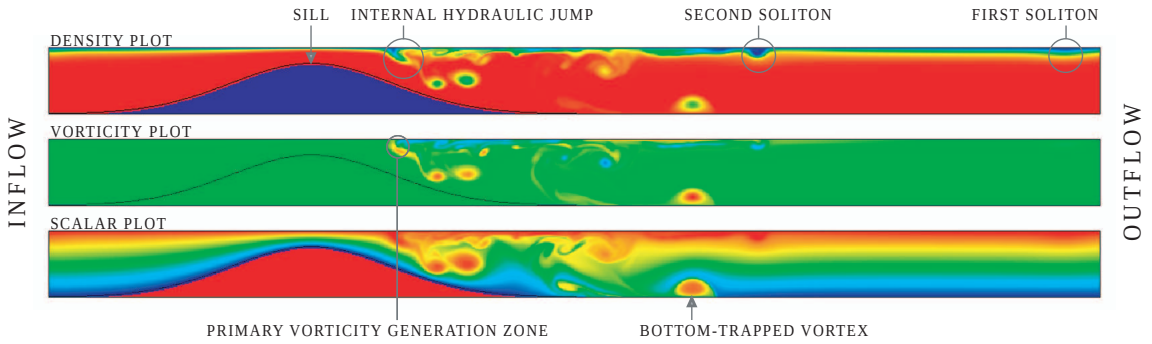

Figure 6.6: Stratified flow past a sill: significant features (at $t = 4750$ $[s]$).

Figures 6.7-6.9 show the evolution of the flow past the sill. Figures 6.10-6.13 show timeseries of integral quantities (i.e. integrals over the whole flow domain) of mass and energy for this calculation. An internal hydraulic jump forms at the downstream side of the sill, as can be seen in the density plots (Figure 6.7). The bifurcation point (at the hydraulic jump in Figure 6.6), where the strong downslope flow detaches from the surface remains relatively stationary throughout the calculation. The downslope currents entrain lighter waters from the "stagnant" pool just downstream of the bifurcation point, with entrained lighter waters advected downstream at the cores of vortices. The vortices are generated at the shear layer downstream of the bifurcation point (see frame 4 in Figure 6.8).

Interestingly, three coherent structures are ejected from the hydraulic jump region. First, a low frequency soliton propagates as a wave of depression along the pycnocline (region of rapidly changing density) and exits the domain at $t \approx 5000$ seconds (labeled "First soliton" in Figure 6.6). This is easily identified as the large mass spike in figure 6.13, because the outflowing profile has less mass (i.e. lower density) than the inflowing profile. This is also apparent in the second to last frame in Figure 6.7. This low frequency wave is

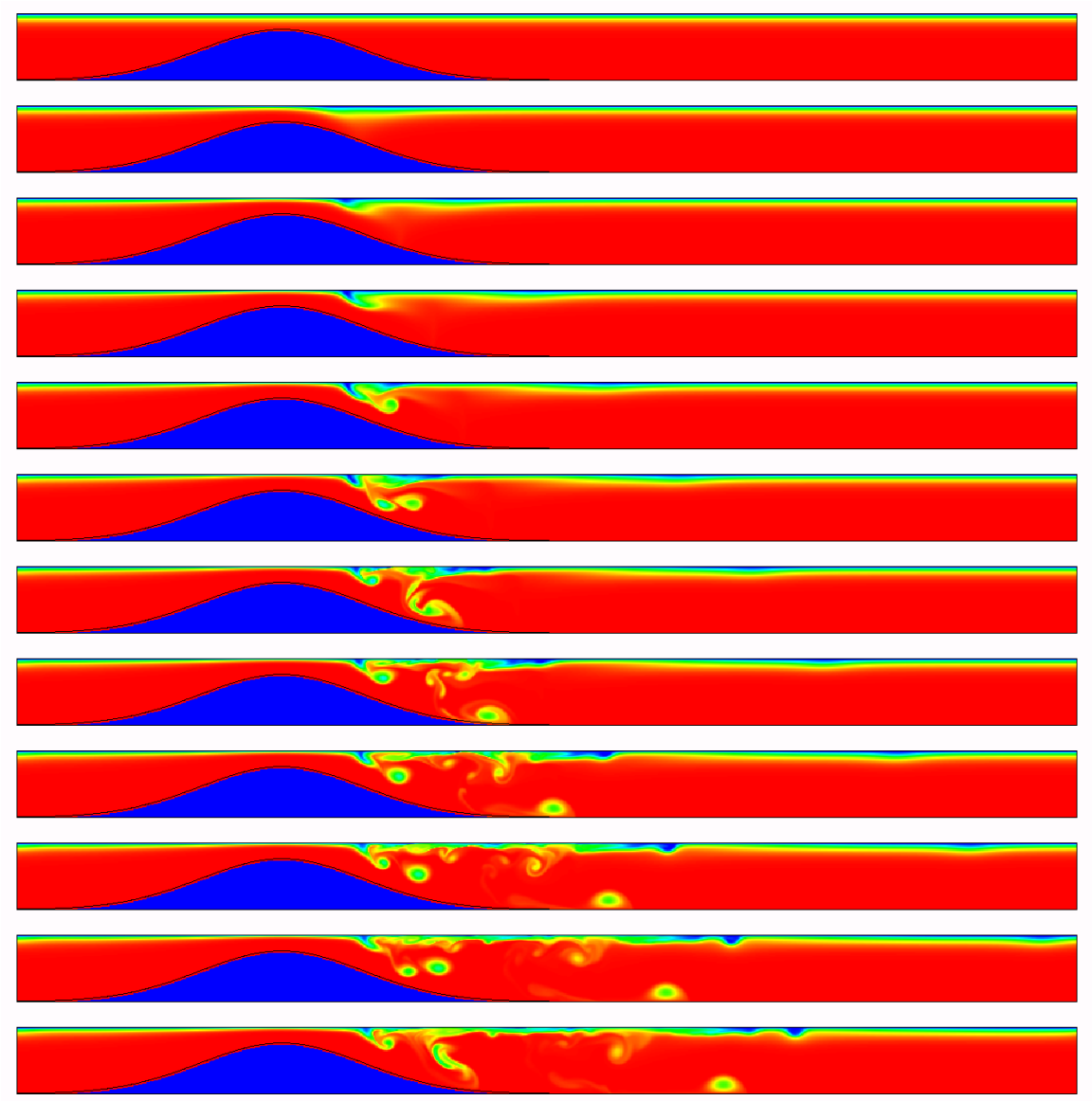

Figure 6.7: Stratified flow past a sill. Density plots, with 15.833 minutes between frames. ($t = [0, 475, 950, 1425, 1900, 2375, 2850, 3325, 3800, 4275, 4750, 5225]$ seconds). Red is 1001 $[kg/m^3]$, blue is 1000 $[kg/m^3]$.

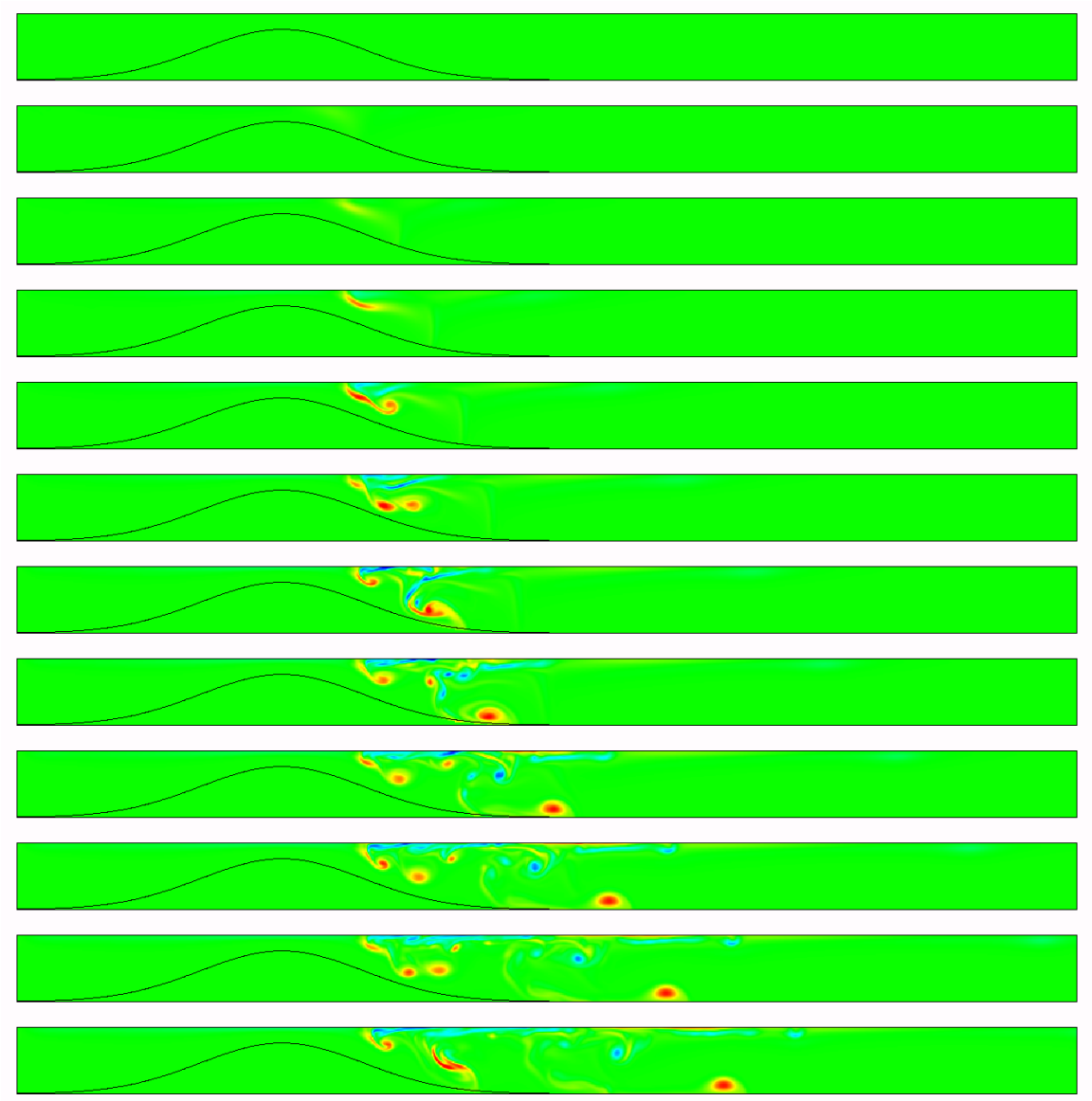

Figure 6.8: Stratified flow past a sill. Vorticity plots, with 15.833 minutes between frames ($t = [0, 475, 950, 1425, 1900, 2375, 2850, 3325, 3800, 4275, 4750, 5225]$ seconds). Red is counter-clockwise vorticity, blue is clockwise. The vorticity range is $\approx \pm 0.05$ $[1/s]$.

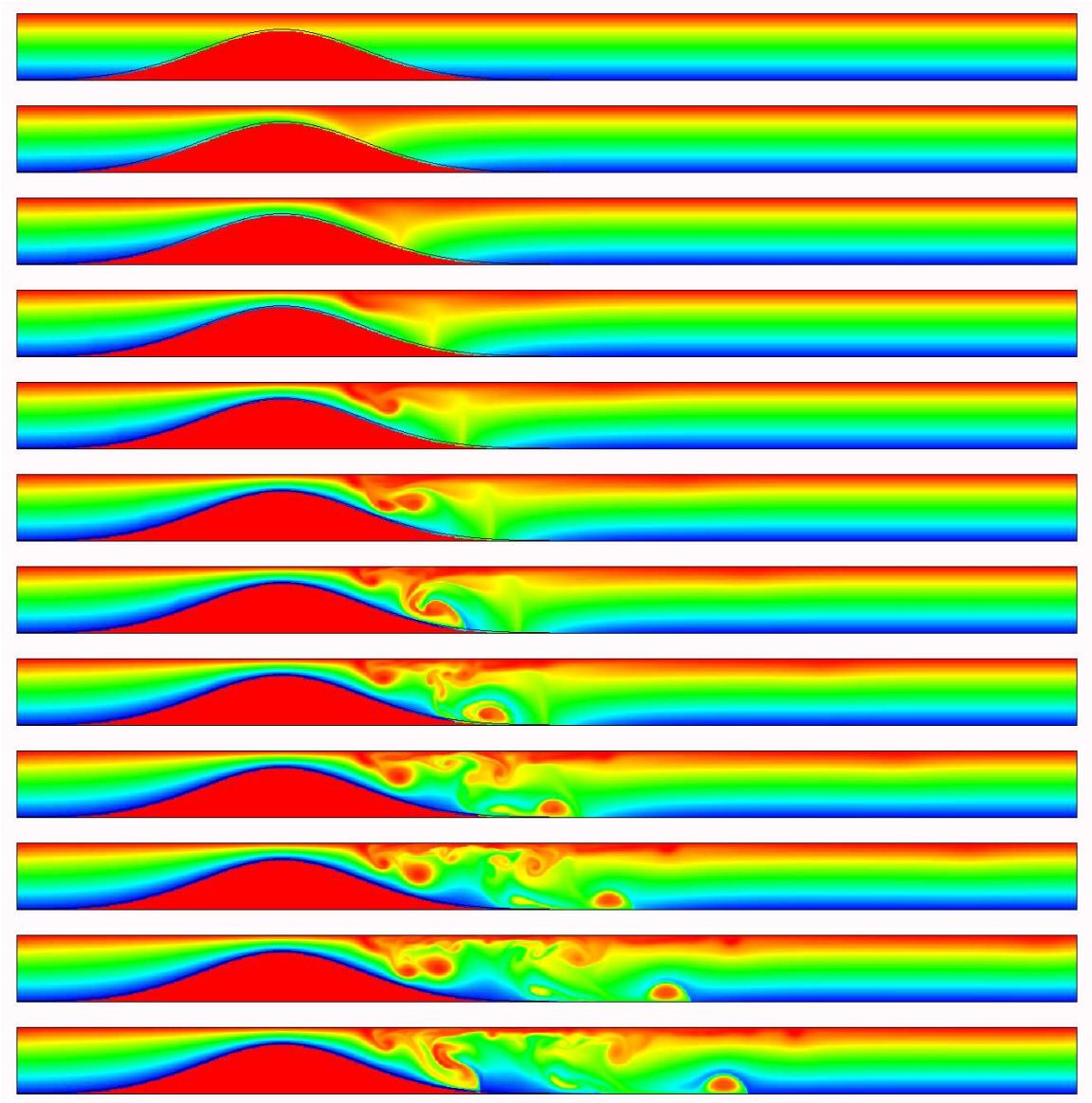

Figure 6.9: Stratified flow past a sill. Passive scalar plots, with 15.833 minutes between frames. ($t = [0, 475, 950, 1425, 1900, 2375, 2850, 3325, 3800, 4275, 4750, 5225]$ seconds). Initially, red is $z = 0$, blue is $z = -256$ meters with a linear distribution between.

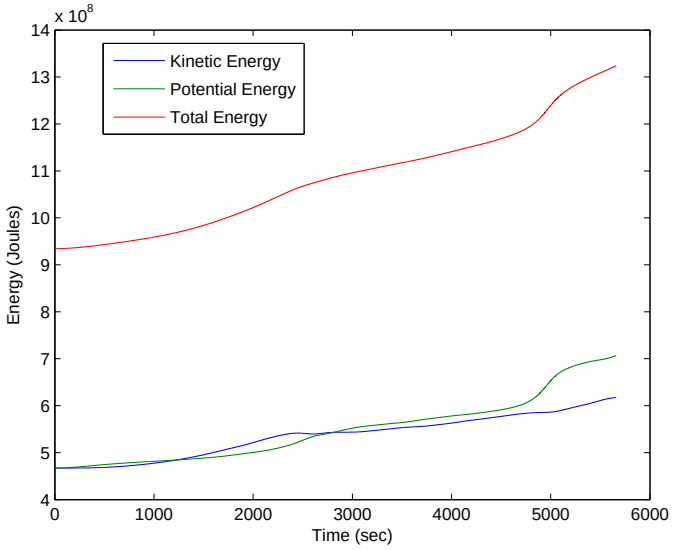

Figure 6.10: Stratified flow past a sill. Kinetic, potential and total energy timeseries.

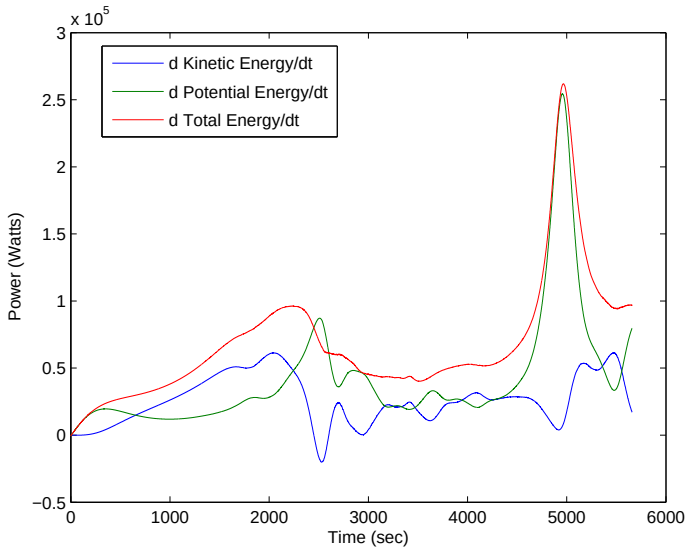

Figure 6.11: Stratified flow past a sill. Change in kinetic, potential and total energy timeseries.

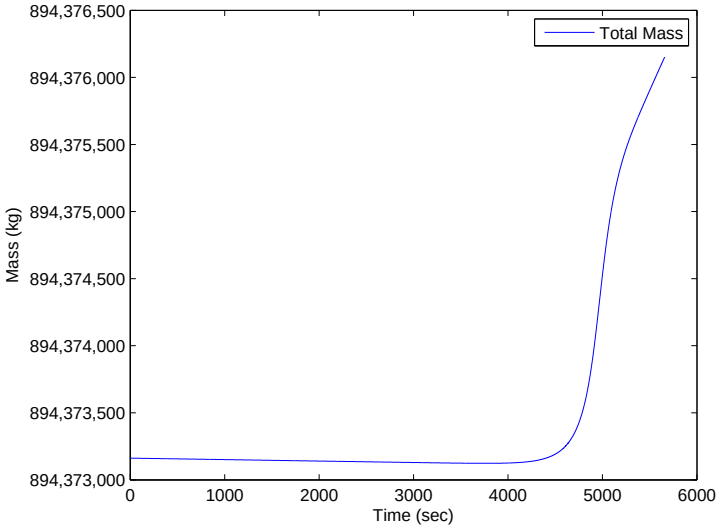

Figure 6.12: Stratified flow past a sill. Total mass timeseries.

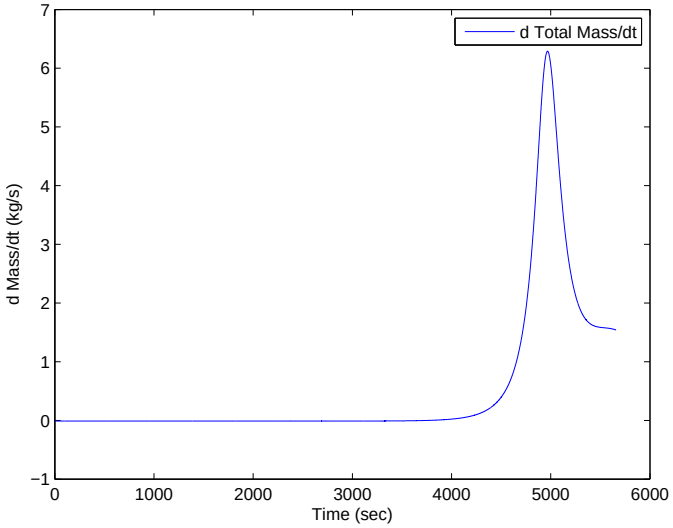

Figure 6.13: Stratified flow past a sill. Mass flux timeseries.

also clearly evident in the density plot, yet it's structure is not clear in the vorticity plot due to it's efficient propagation. Next a shorter wavelength, solitary wave of depression is ejected and propagates downstream along the pycnocline (labeled "second soliton" in Figure 6.6). Finally, a large vortex patch is shed from the shear layer at the internal hydraulic jump and propagates downstream along the bed (labeled "bottom-trapped vortex" in Figure 6.6). This vortex core has lighter waters in it's core, but is trapped along the bottom due to pressure gradients that are stronger than the buoyancy forcing. The vorticity plots also show the merging (see frames 6-7) of vortex patches that are rotating in the same direction.

A careful inspection of the timeseries plots validates these findings. Specifically, the system maintains essentially constant mass (see Figure 6.13) until $t \approx 4000$ seconds when the leading edge of the first depression wave begins to hit the outflow face. The spike in potential energy change at $t \approx 2500$ seconds (and the other spikes before $t = 4000$) are not due to outflowing mass, but to an exchange with kinetic energy, i.e. heavy water is lifted due to vertical velocity at the hydraulic jump. The gradual increase in kinetic energy is due to the trapping of kinetic energy in the jump region, while the gradual increase in potential energy is due to both the lifting of heavy water over the sill (notice the first 200 seconds in Figure 6.11 where nearly all energy is spent lifting the fluid over the sill), and the outflow of lighter waters relative to the inflowing waters. Eventually, we expect the system to reach a dynamic equilibrium state, with continued vortex shedding, entrainment of lighter waters and internal wave generation. Note that the initial conditions are idealized and the flow is likely highly 3D in realistic oceanic settings. Nonetheless, we expect similar features to develop for similar flows past sills.

A convergence study (using 16 processors) for the first 4 minutes of this problem is presented in Tables 6.25-6.27. This shows that we are getting the expected accuracy for this complex field-scale variable density flow.

Table 6.25: Solution error convergence rates using L_∞ norm: $\Delta x_c = \frac{1}{256} = 2\Delta x_m = 4\Delta x_f$; $\Delta x_c = (1.6000e+01, 1.6000e+01) = 2\Delta x_m = 4\Delta x_f$; a $2D$ calculation. Isotropic mesh spacing.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	1.1622e-02	5.9232e-03	**9.7246e-01**
V-Velocity	1.0837e-02	5.7026e-03	**9.2630e-01**
Pressure	4.0891e-03	2.5460e-03	**6.8354e-01**
Density	5.7153e-02	1.9954e-02	**1.5182e+00**
Scalar-0	5.9958e+00	2.9695e+00	**1.0137e+00**
Scalar-1	6.0098e+00	3.0724e+00	**9.6796e-01**
p_x/ρ	1.0151e-04	5.0116e-05	**1.0183e+00**
p_y/ρ	2.8738e-04	1.5055e-04	**9.3278e-01**
Z-Vorticity	5.6106e-04	2.2425e-04	**1.3231e+00**

Table 6.26: Solution error convergence rates using L_1 norm: $\Delta x_c = \frac{1}{256} = 2\Delta x_m = 4\Delta x_f$; $\Delta x_c = (1.6000e+01, 1.6000e+01) = 2\Delta x_m = 4\Delta x_f$; a $2D$ calculation. Isotropic mesh spacing.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	6.3982e-05	1.6330e-05	**1.9701e+00**
V-Velocity	4.7190e-05	1.3562e-05	**1.7989e+00**
Pressure	3.3547e-04	8.8763e-05	**1.9182e+00**
Density	2.6976e-03	6.6357e-04	**2.0233e+00**
Scalar-0	4.2447e-02	1.2221e-02	**1.7963e+00**
Scalar-1	4.0836e-02	1.1791e-02	**1.7922e+00**
p_x/ρ	6.7210e-07	2.1273e-07	**1.6597e+00**
p_y/ρ	1.1772e-06	2.7423e-07	**2.1019e+00**
Z-Vorticity	5.0836e-06	2.0216e-06	**1.3304e+00**

Table 6.27: Solution error convergence rates using L_2 norm: $\Delta x_c = \frac{1}{256} = 2\Delta x_m = 4\Delta x_f$; $\Delta x_c = (1.6000e+01, 1.6000e+01) = 2\Delta x_m = 4\Delta x_f$; a $2D$ calculation. Isotropic mesh spacing.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
U-Velocity	2.4874e-04	8.1063e-05	**1.6175e+00**
V-Velocity	3.3952e-04	1.2351e-04	**1.4589e+00**
Pressure	5.2176e-04	2.1079e-04	**1.3076e+00**
Density	7.2390e-03	1.8842e-03	**1.9419e+00**
Scalar-0	2.0823e-01	7.5751e-02	**1.4589e+00**
Scalar-1	3.2061e-01	1.2333e-01	**1.3783e+00**
p_x/ρ	3.5105e-06	1.1150e-06	**1.6547e+00**
p_y/ρ	5.9181e-06	1.4459e-06	**2.0332e+00**
Z-Vorticity	2.2425e-05	9.4235e-06	**1.2507e+00**

6.5.3 Internal Wave Dissipation On a Uniform Slope

Here we test our ability to simulate internal waves breaking on a slope using DWPM1. For this test case we follow the lab-scale experiments of [77] and simulate lab scale dissipation of a solitary internal wave on a slope. The experimental tank is 3 [m] long by 0.5 [m] wide and tall, with the 8:1 (vertical:horizontal) slope meeting the vertical wall of the tank half way up. We initialize the density field as in figure 6.19, with salt water, $\rho = 1030$ $[kg/m^3]$, on the bottom and fresh water, $\rho = 1000$ $[kg/m^3]$, on the top, and a perturbed interface following $\phi = -0.25 - 0.15e^{-3x^2}$. We smooth the interface over 0.1 [m] using the heaviside smoothing kernal (equation 60 in [99]).

We present our highly resolved 2D AMR calculations in the time sequence of figures 6.14-6.18. We present our 3D single-level calculations in the time sequence of figures 6.19-6.24. In these figures, blue indicates fresh water and red is for salt water. In the 2D figures we included the AMR disjoint box outlines. In the 3D figures we included an isosurface of the density interface and streamtubes to aid in visualizing the simulation.

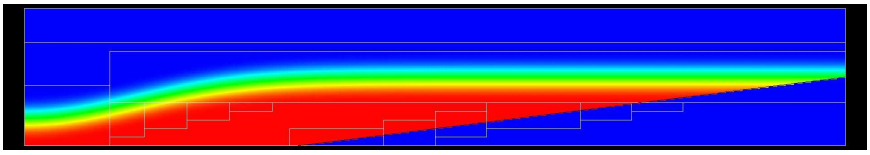

Figure 6.14: Internal wave breaking 2D: initial conditions density plot. Blue is saltwater, red is freshwater. Boxes indicate refined regions.

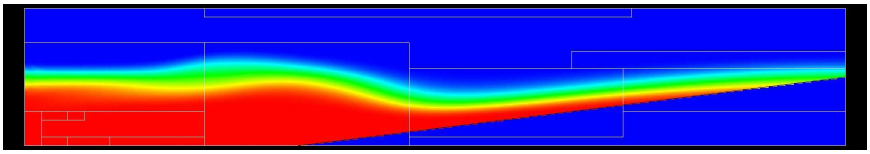

Figure 6.15: Internal wave breaking 2D: propagation density plot. Blue is saltwater, red is freshwater. Boxes indicate refined regions.

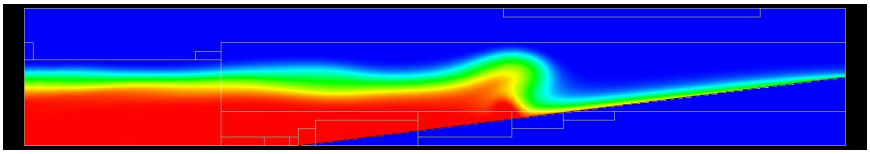

Figure 6.16: Internal wave breaking 2D: shoaling density plot. Blue is saltwater, red is freshwater. Boxes indicate refined regions.

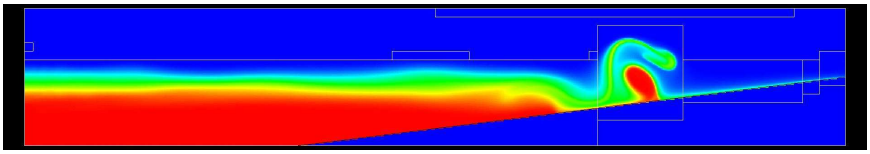

Figure 6.17: Internal wave breaking 2D: breaking density plot. Blue is saltwater, red is freshwater. Boxes indicate refined regions.

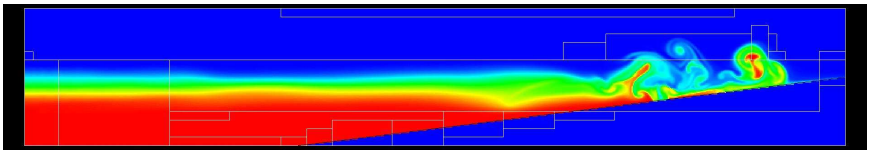

Figure 6.18: Internal wave breaking 2D: dissipation density plot. Blue is saltwater, red is freshwater. Boxes indicate refined regions.

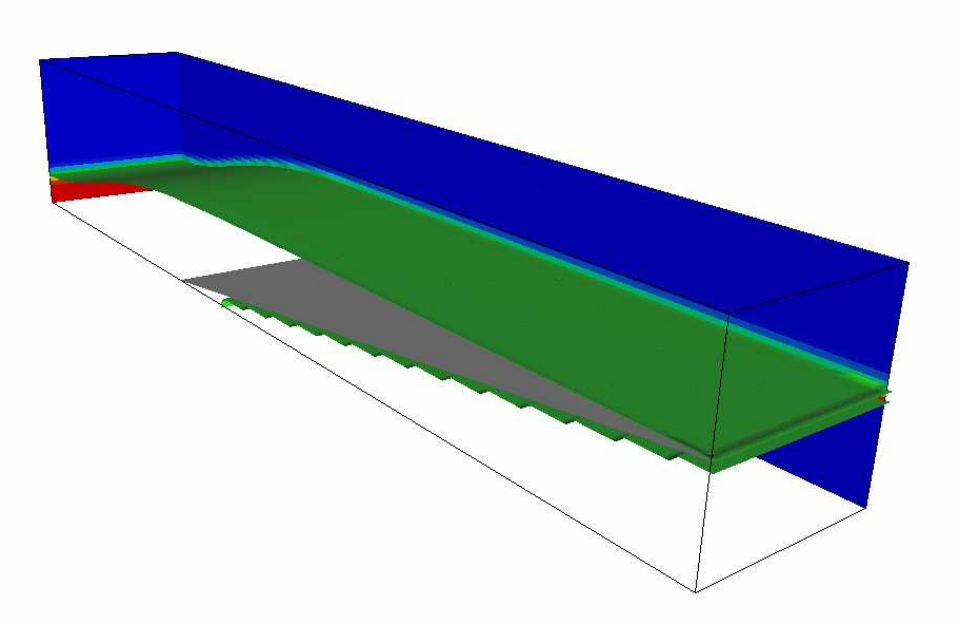

Figure 6.19: Internal wave breaking 3D: initial conditions, $t = 0.00$ $[s]$. Blue is saltwater, red is freshwater. Iso-contour of salt-fresh interface, streamtubes aid in visualizing flow.

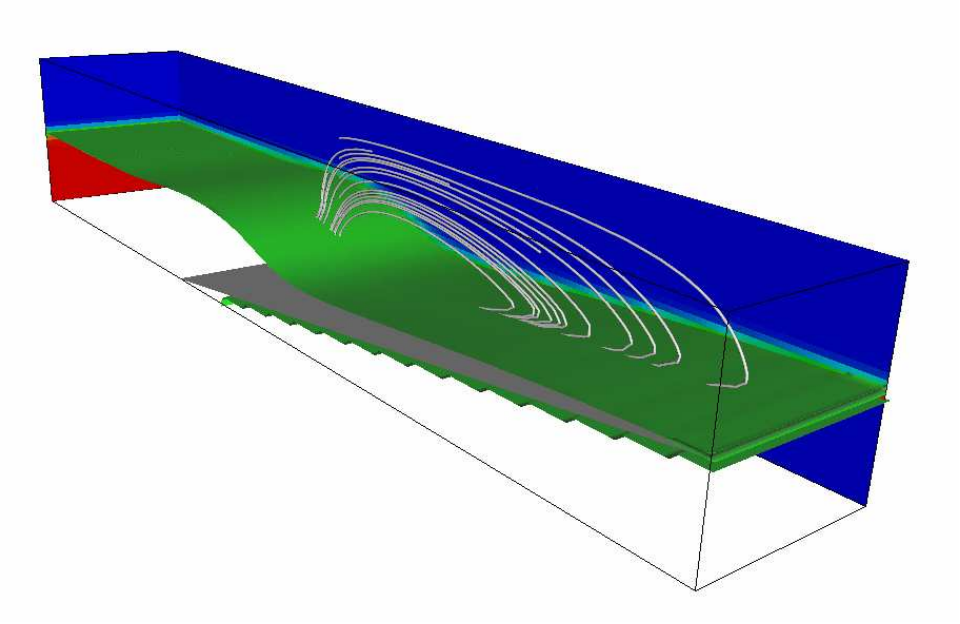

Figure 6.20: Internal wave breaking 3D: shoaling, $t = 13.70$ $[s]$. Blue is saltwater, red is freshwater. Iso-contour of salt-fresh interface, streamtubes aid in visualizing flow.

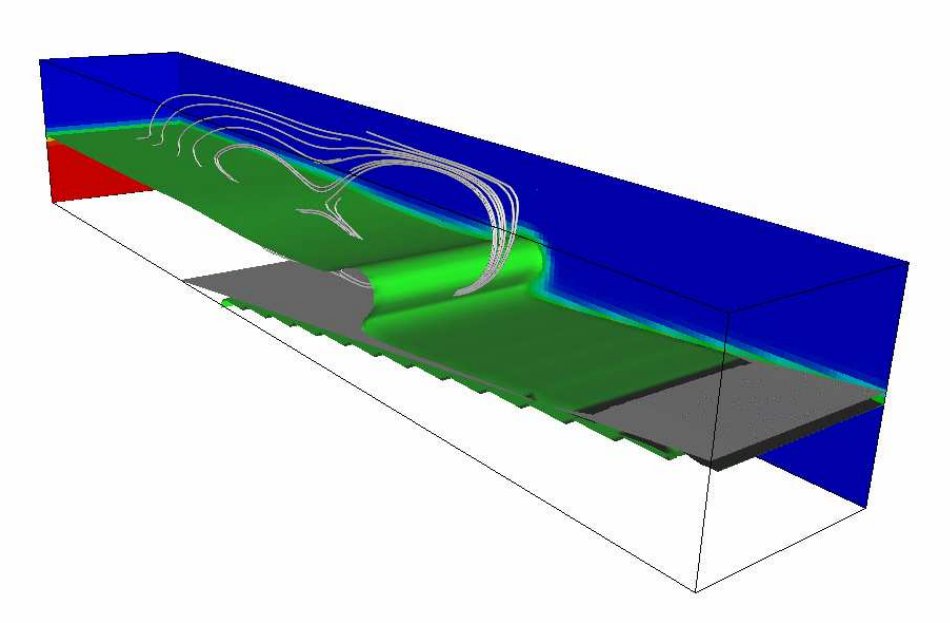

Figure 6.21: Internal wave breaking 3D: breaking, $t = 20.24$ $[s]$. Blue is saltwater, red is freshwater. Iso-contour of salt-fresh interface, streamtubes aid in visualizing flow.

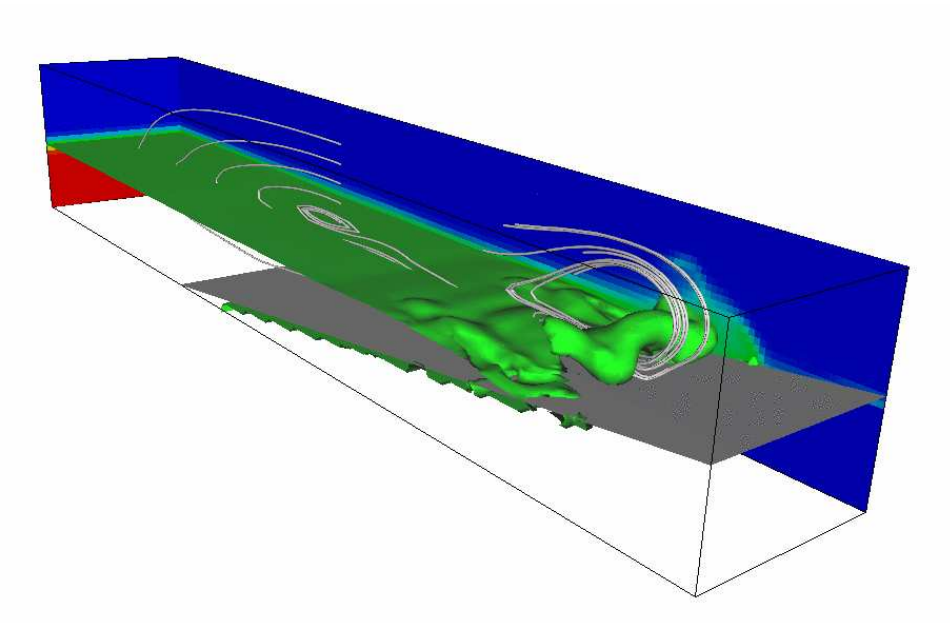

Figure 6.22: Internal wave breaking 3D: breaking, $t = 28.69$ $[s]$. Blue is saltwater, red is freshwater. Iso-contour of salt-fresh interface, streamtubes aid in visualizing flow.

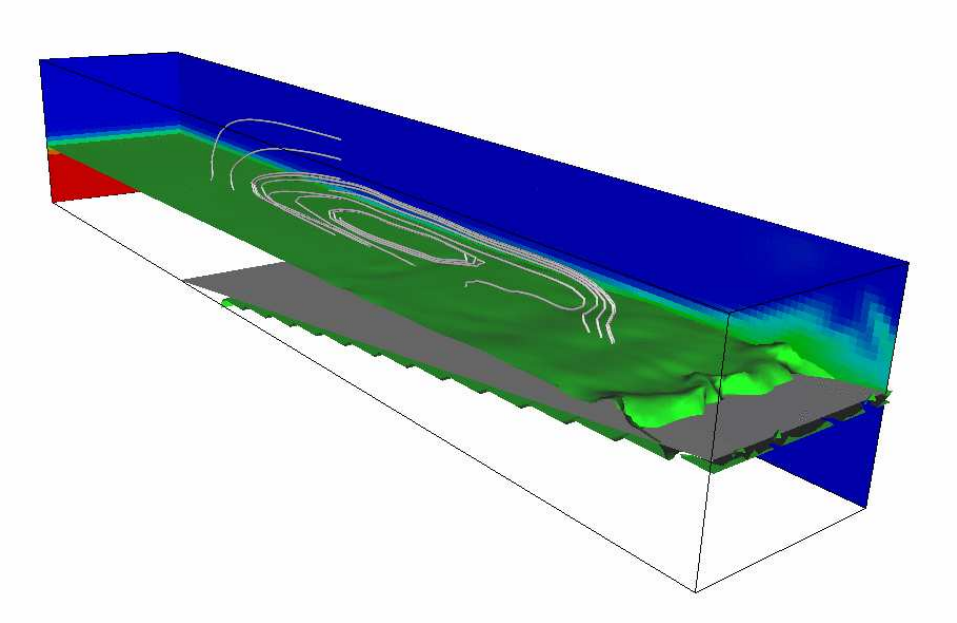

Figure 6.23: Internal wave breaking 3D: dissipation, $t = 38.31$ $[s]$. Blue is saltwater, red is freshwater. Iso-contour of salt-fresh interface, streamtubes aid in visualizing flow.

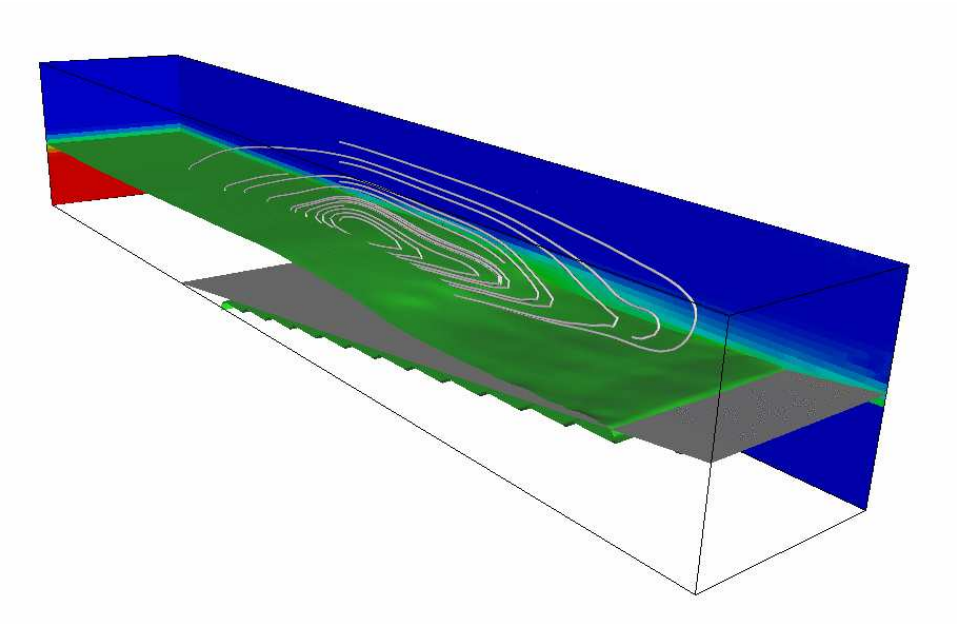

Figure 6.24: Internal wave breaking 3D: dissipation, $t = 49.57$ $[s]$. Blue is saltwater, red is freshwater. Iso-contour of salt-fresh interface, streamtubes aid in visualizing flow.

6.6 Idealized Flow in Lake Tahoe

For a challenging field-scale test of the method, here we calculate idealized flows in Lake Tahoe. The main goal of this Section is to illustrate that our method could be used to address field-scale environmental fluid mechanics questions.

Through remote and direct observations, fascinating upwelling and lake circulation patterns have been identified in Lake Tahoe, CA-NV [98, 97, 91]. These likely play a significant role in the dynamics of Lake Tahoe water quality. These patterns are thought to be caused by wind shear induced surface currents that interact with the density distribution, and the lake bathymetry. Through field data and conceptual analysis, researchers have identified these surface patterns as signatures of upwelling events [98, 97, 91]. These complex upwelling events are poorly understood due to sparse spatial and temporal observations, yet they are observed in many lakes around the world.

Lake Tahoe is so weakly forced that at present the only feasible long term calculations with our non-hydrostatic method are constant density. This is due to the fact that the method is constrained by an advective CFL number that depends on internal wave speeds, which can be on the order of 1 $[m/s]$ for Lake Tahoe [90]. During "spin up" of the model the wind forcing is so light that intial maximum velocities are on the order of 10^{-9} $[m/s]$. This means that given todays computational resources, the "spin up" duration is not feasible for this version of our method. We anticipate further optimizations, and advances in computer hardware that will enable realistic non-hydrostatic 3D calculations in the not too distant future.

Our first idealized Lake Tahoe test is flow in a 2D, East-West, cross-section of

the lake, with constant density (to avoid the internal wave CFL constraint during "spin up"). The cross-section is taken from a USGS DEM of the lake, and is cut to pass through Deadman Point on the East side of the lake, as is indicated in Figure 6.25. Our 2D grid has a 17 $[m]$ isotropic mesh spacing. We force the model for 12 days with a constant 10 $[m/s]$ wind blowing to the East. This simulation is shown in Figure 6.26, with a zoom in shown in Figure 6.27. As expected, the model (using DWPM1 of Section 4.1.4) simulates a surface layer moving to the East. When this advancing surface layer encounters the East shoreline it plunges and recirculates as a slow moving jet that interacts dramatically with the bottom topography. The jet is clearly seen in the zoomed in Figure 6.27. On the West side of the Figure 6.26 cross-section we note an upwelling return current that feeds the surface layer. Note that the timescale over which this happens is 12 days, with maximum interior (non-surface currents) jet velocities in the range of $2-4$ $[cm/s]$.

While these images look impressive, it is important to note that Lake Tahoe is in fact thermally stratified (although it can be well mixed to a depth of 300-500 $[m]$ in late winter), and therefore such a jet would not penetrate as deep into the lake (not to mention 3D effects). Nonetheless, researchers might find similar slow moving jets along the pycnocline.

Our next test is to calculate 3D flow in Lake Tahoe. We force the model with the same constant wind as the 2D case. The 3D grid has an 80 $[m]$ isotropic mesh spacing. For this simulation our preliminary results in Figure 6.28 indicate three dimensional flow features. In this calculation we are severely computationally limited (even on 128 processors of a DOE supercomputer), and therefore are only able to simulate the initial period of the

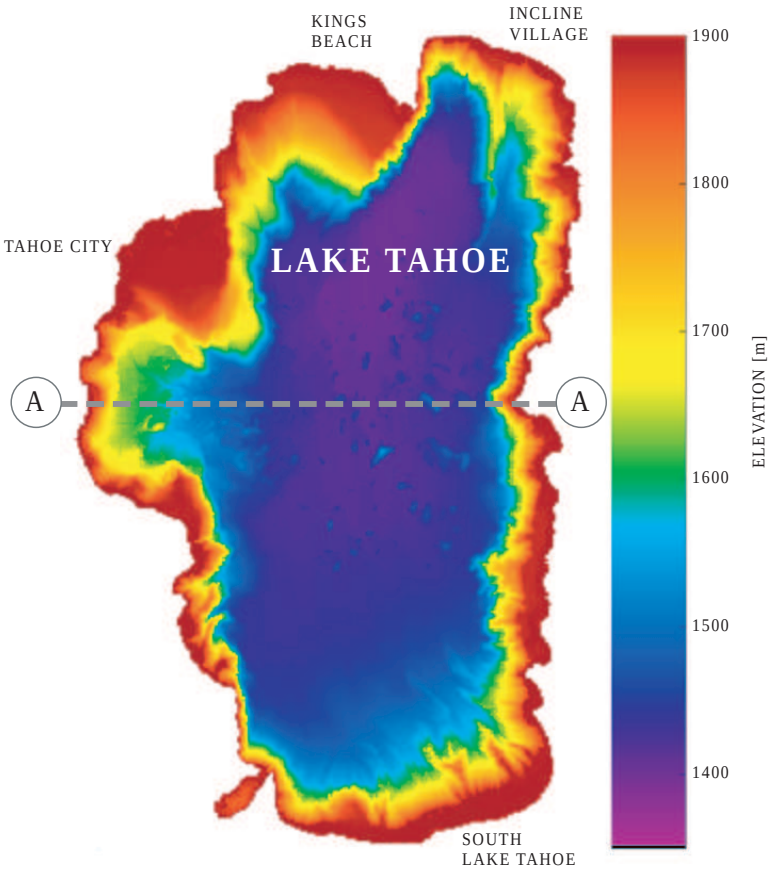

Figure 6.25: Lake Tahoe bathymetry. Section A-A indicates the location of the 2D calculations. Elevations are relative to sea level.

constant density "spin up", roughly the equivalent of frame 2 in Figure 6.26. It is interesting to note that in figure 6.28, we are seeing 3D motions in the results. Notice the gyre near South Lake Tahoe, and the overall overturning rotational motion.

Recall the disclaimer that this is a constant density calculation, and neglects stratification and rotational effects. These Lake Tahoe tests are not meant to reproduce actual conditions. Stratification plays a significant role in the motions within Lake Tahoe. For a hydrostatic approximation to flows in Lake Tahoe the reader may find [90] of interest. This Section is only meant to illustrate our ability to simulate flows in realistic geometry at field-scale.

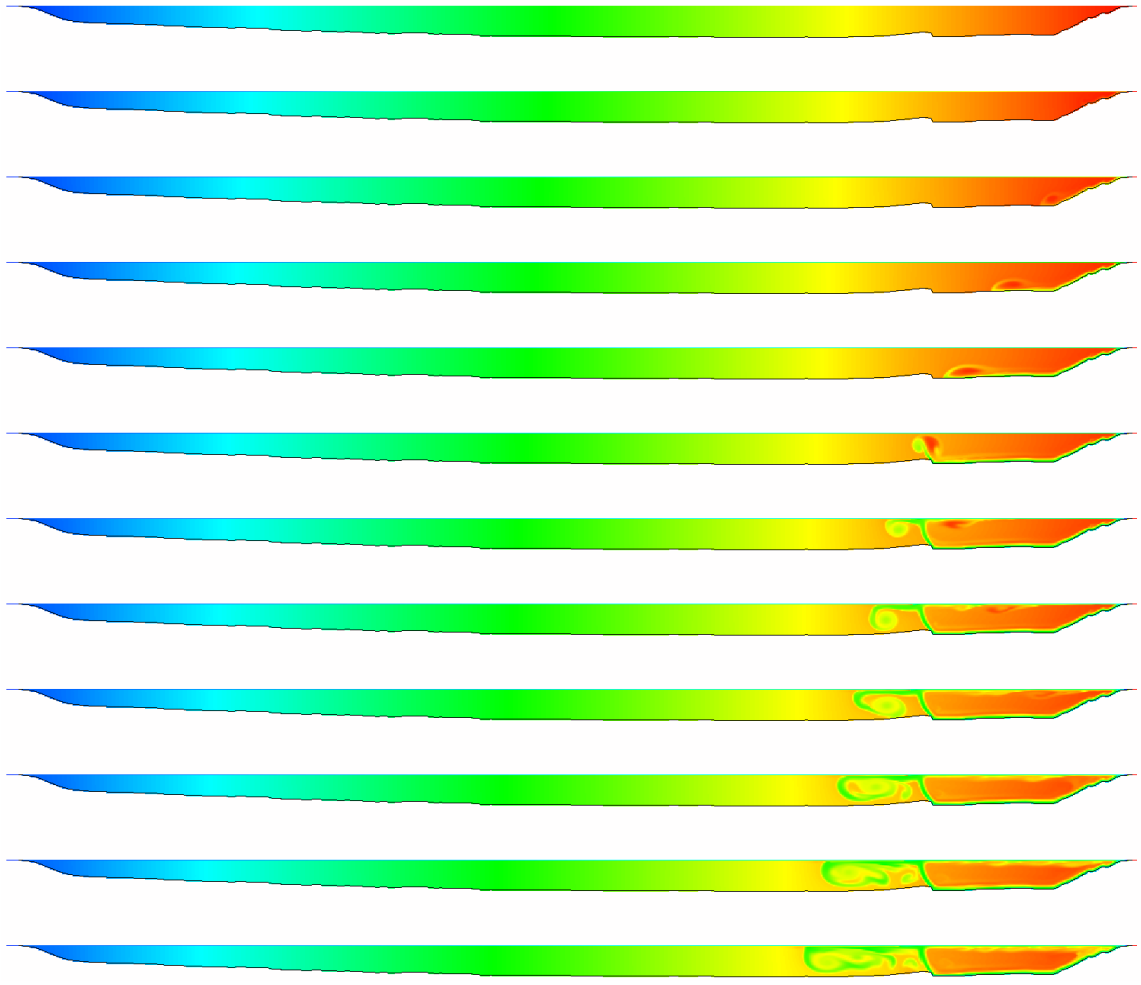

Figure 6.26: Idealized 2D Lake Tahoe circulation, a passive scalar time sequence at $t = [0.00, 2.47, 3.88, 4.92, 5.78, 6.61, 7.43, 8.26, 9.14, 10.04, 10.93, 11.82]$ days. Wind forcing is from left to right. Color coding is a passive scalar advected with the flow.

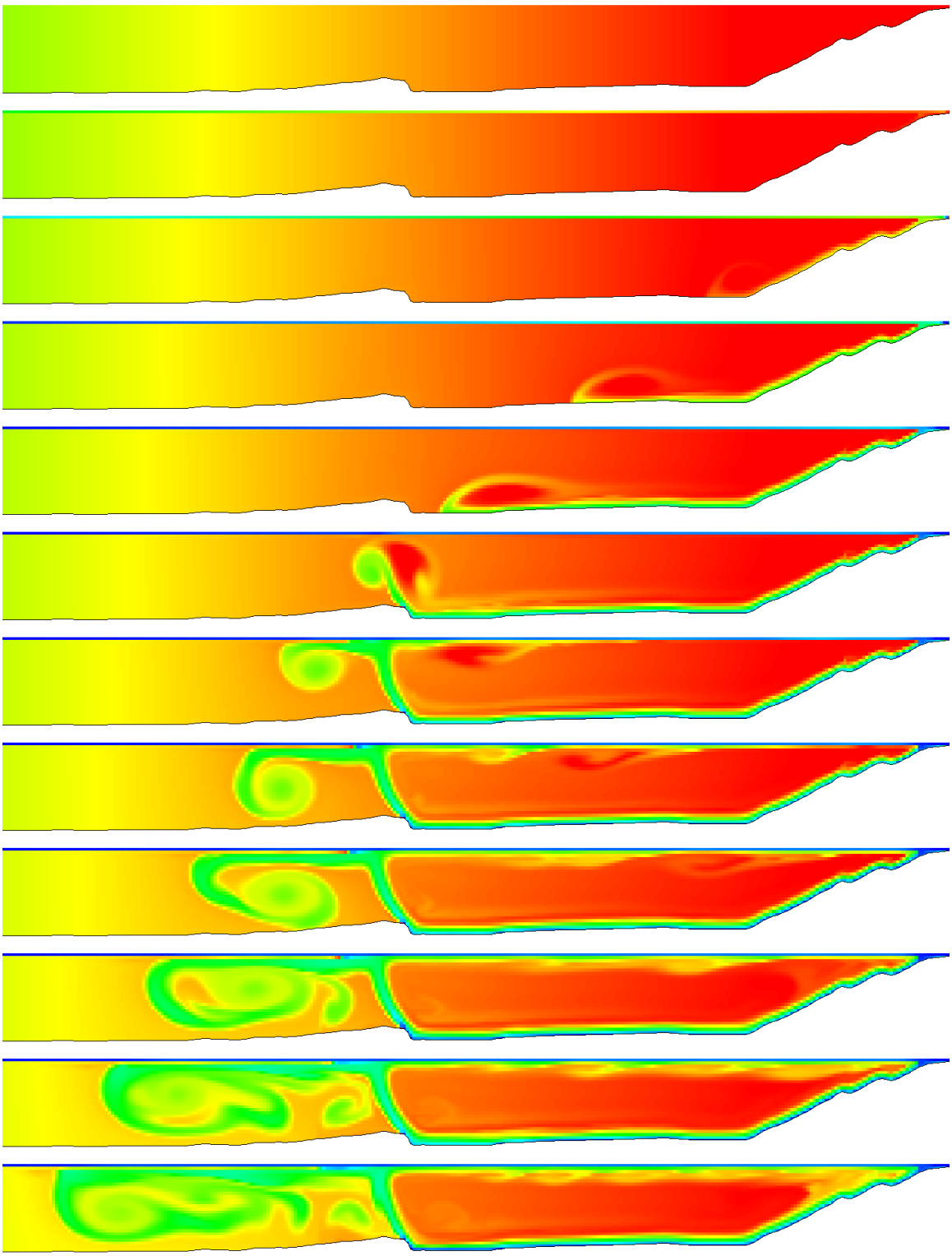

Figure 6.27: Idealized 2D Lake Tahoe circulation, a passive scalar time sequence at $t = [0.00, 2.47, 3.88, 4.92, 5.78, 6.61, 7.43, 8.26, 9.14, 10.04, 10.93, 11.82]$ days. Zoomed in on the right side of Figure 6.26, and with a more constrained color-map.

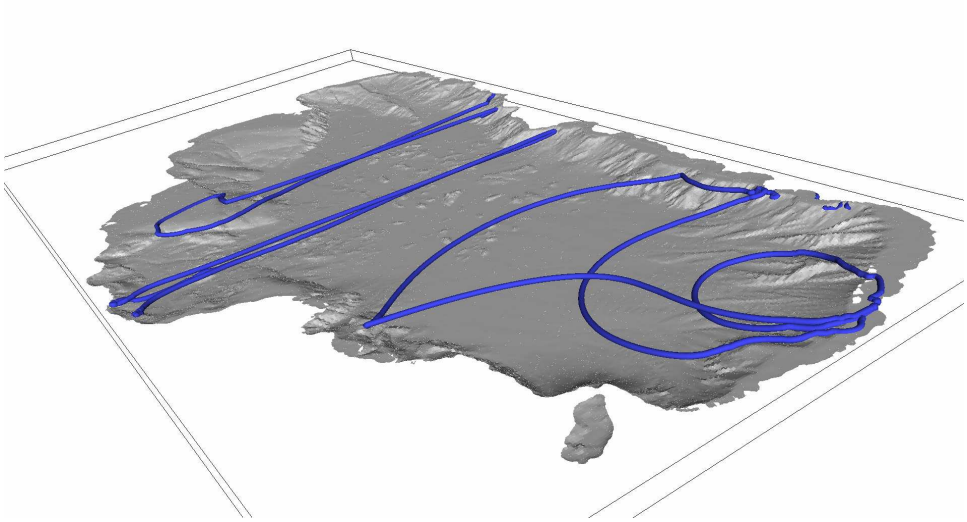

Figure 6.28: Idealized 3D Lake Tahoe circulation. Streamlines indicate instantaneous current direction for a hypothetical constant density calculation.

Chapter 7

Conclusion

7.1 Summary and Conclusions

This thesis presents an adaptive Cartesian grid projection method that is suitable for the accurate numerical study of incompressible fluids.

First our embedded boundary discretization was presented, including the geometric description, and discretizations of elliptic, parabolic and hyperbolic PDEs. The geometric description is relatively new, and is efficient and robust for all problems we have tried. This is because the geometry is generated in $O(N^{\frac{D-1}{D}})$ calculations from easily defined distance functions, is "water-tight" and therefore allows exactly conservative finite volume discretizations, handles arbitrarily complex geometries, and is easily coarsened/refined for use in multigrid and AMR.

Second our single-grid finite volume discretization of the incompressible Navier-Stokes equations were presented. This algorithm for solving the incompressible Navier-Stokes equations in complex geometry with buoyancy forcing, extends the work of [25, 26,

27, 9, 12, 1, 72] while maintaining second-order accuracy.

Third we extended the single-grid algorithm to block-structured adaptive mesh refinement. This method enables the efficient and accurate simulation of multiscale, environmental flows in realistic complex geometry. For example, the greatest impediment to the numerical simulation of oceanic internal waves is the broad range of length scales over which they exist. This is not surprising for most field-scale numerical simulations especially when dissipation scales are involved. However, internal waves still present a challenging computational problem simply due to the range of scales which govern their generation, propagation, and breaking even if all the dissipative scales are not to be captured. AMR makes the simulation of multiscale highly nonlinear internal waves a tractable problem.

Fourth we presented results from a range of test cases to validate that our method is in fact second-order accurate. Our three lab-scale tests (spherical driven cavity, trapped vortex, and flow past a sphere) show that our method is performing as expected. The flow past a sphere example also validated our adaptive algorithm. To test the density-weighted projection methods we simulated field-scale stratified flow past a sill, and breaking internal waves on a lab-scale slope, with and without AMR. For the flow past a sill test we validated that our method is second-order accurate, and produced some interesting results (an internal hydraulic jump, vortex shedding, bottom-trapped vortices, and internal solitary waves of depression). Finally we tested the method using a realistic field-scale bathymetry, Lake Tahoe. The Lake Tahoe simulations are highly idealized and illustrated a need for a code optimization cycle. The Lake Tahoe simulations are so weakly forced by wind stress that the "spin up" time for cases including stratification is prohibitive.

7.2 Future Work

Our first task for future work is to implement optimizations to speed up the code. Serial and parallel performance optimizations are clearly needed in the elliptic solver. Optimizations of our current solver may include reducing the communication costs by a) utilizing additional ghost-cells, and b) improving our load-balancing algorithm. Though we use multigrid, an $O(N)$ method, the communication costs on parallel computers is a limiting factor to parallel scaling. We might also need to take a more significant approach to speeding up our elliptic solver, and derive a cell-centered EB AMR approach following [76].

Extension of this work to the case where the coarse-fine interface is allowed to cross the embedded boundary interface would be useful. This would remove the constraint that all irregular control-volumes have maximum refinement (the hyperbolic case has already been successfully addressed in [31]). The elliptic case is relatively straightforward, and at the interface crossing requires modification of: 1) the Dirichlet embedded boundary condition, 2) the coarse-fine interpolation step and 3) conservative coarse-fine refluxing. An initial implementation of the coarse-fine crossing the EB interface for the elliptic algorithm is promising.

An additional useful extension is the ability to subcycle the levels to maintain consistent CFL numbers across all control volumes. This extension would follow the works of [1, 72], with an additional freestream conservation fix following [33].

We anticipate extending the method to higher-order accuracy by following the work of [5]. The quadrature formulas in [5] provide a systematic mechanism for distinguishing between averages over cells, averages over faces, and point values, to fourth-order accuracy.

This can be combined with the ideas here and in [34] to obtain fourth-order in space finite-volume discretizations for nonlinear hyperbolic problems on a locally-refined grid. It is not obvious how to extend the Mehrstellen discretizations in [5] to the case where the right-hand side includes a time derivative, particularly in the case where implicit differencing in time is required. We will consider a variety of possible approaches here, including fully implicit methods and predictor-corrector approximations to such methods in which the Mehrstellen correction is treated explicitly. The fourth-order spatial approach is straightforward to pursue in conjunction with second-order accurate temporal discretizations. However, the extension to higher order in time is still an active research issue [18]. It would be necessary to compute higher moments of the intersections between the irregular domain and the Cartesian grid.

Bibliography

[1] A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome. A conservative adaptive projection method for the variable density incompressible Navier-Stokes equations. *J. Comp. Phys.*, 142:1–46, 1998.

[2] A. S. Almgren, J. B. Bell, and W. Y. Crutchfield. Approximate projection methods: Part i. inviscid analysis. Technical Report LBNL-43374, LBNL, May 1999.

[3] J. R. Apel. A new analytical model for internal solitons in the ocean. *J. Phys. Oceanogr.*, 33:2247–2269, 2003.

[4] J. R. Apel, J. R. Holbrock, A. K. Liu, and J. J. Tsai. The sulu sea internal soliton experiment. *J. Phys. Oceanogr.*, 15:1625–1651, 1985.

[5] M. F. Barad and P. Colella. A fourth-order accurate adaptive mesh refinement method for Poisson's equation. *J. Comp. Phys.*, 209(1):1–18, October 2005.

[6] M. F. Barad, P. Colella, and D. T. Graves. An adaptive Cartesian grid embedded boundary method for the incompressible Navier-Stokes equations. *J. Comp. Phys.* In prep.

[7] M. F. Barad, P. Colella, D. T. Graves, T. J. Ligocki, P. O. Schwartz, D. Trebotich, and B. VanStraalen. A Cartesian grid embedded boundary method for the incompressible Navier-Stokes equations. *J. Comp. Phys.* In prep.

[8] J. Bell, M. Berger, J. Saltzman, and M. Welcome. Three-dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM J. Sci. Comput.*, 15(1):127–138, 1994.

[9] J. B. Bell, P. Colella, and H. M. Glaz. A second-order projection method for the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 85:257–283, 1989.

[10] J. B. Bell, M. S. Day, C. A. Rendleman, S. E. Woosley, and M. A. Zingale. Adaptive low mach number simulations of nuclear flames. *J. Comp. Phys.*, 195:677–694, 2004.

[11] J. B. Bell, M. S. Day, I. G. Shepherd, M. R. Johnson, R. K. Cheng, J. F. Grcar, V. E. Beckner, and M. J. Lijewski. From The Cover: Numerical simulation of a laboratory-scale turbulent V-flame. *Proceedings of the National Academy of Sciences*, 102(29):10006–10011, 2005.

[12] J. B. Bell and D. L. Marcus. A second-order projection method for variable-density flows. *Journal of Computational Physics*, 101:334–348, August 1992.

[13] J. B. Bell and G. R. Shubin. An adaptive grid finite difference method for conservation laws. *J. Comp. Phys.*, 52:3:569–591, 1983.

[14] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82(1):64–84, 1989.

[15] M. J. Berger and R. J. Leveque. An adaptive Cartesian mesh algorithm for the euler equations in arbitrary geometries. *AIAA Paper 89-1930CP*, pages 1–7, 1989.

[16] M. J. Berger and J. E. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. Technical report, Stanford, CA, USA, 1983.

[17] M. J. Berger and I. Rigoutsos. An algorithm for point clustering and grid generation. *IEEE Transactions Systems, Man, and Cybernetics*, 21(5):1278–1286, 1991.

[18] A. Bourlioux, A. T. Layton, and M. L. Minion. High-order multi-implicit spectral deferred correction methods for problems of reacting fluid flow. *J. Comput. Phys.*, 189:651–675, 2003.

[19] William L. Briggs, Van Emden Henson, and Steve F. McCormick. *A multigrid tutorial: second edition.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[20] A. N. Brooks and T. J. R. Hughes. Streamline upwind/petrov-galerkin formulations for convection dominated flows with particular emphasis on the incompressible navier-stokes equations. *Comput. Methods Appl. Mech. Eng.*, pages 199–259, 1990.

[21] David L. Brown, Ricardo Cortez, and Michael L. Minion. Accurate projection methods for the incompressible navierstokes equations. *J. Comput. Phys.*, 168(2):464–499, 2001.

[22] D. Cacchione and C. Wunsch. Experimental study of internal waves over a slope. *J. Fluid Mech.*, 66:223–239, 1974.

[23] G. S. Carter, M. C. Gregg, and R-C. Lien. Internal waves, solitary waves, and mixing on the monterey bay shelf. *Continental Shelf Research*, 2004. submitted.

[24] W. Choi and R. Camassa. Fully nonlinear internal waves in a two-fluid system. *J. Fluid Mech.*, 396:1–36, 1999.

[25] A. J. Chorin. Numerical solution of incompressible flow problems. *Studies in Numerical Analysis*, 2:64–71, 1968.

[26] A. J. Chorin. Numerical solutions of the Navier-Stokes equations. *Math. Comp.*, 22:745–762, 1968.

[27] A. J. Chorin. On the convergence of discrete approximations to the Navier-Stokes equations. *Math. Comp.*, 23:341–353, 1969.

[28] A. J. Chorin. Numerical study for slightly viscous flow. *J. Fluid Mech.*, 57:785–796, 1973.

[29] A. J. Chorin. A numerical method for solving incompressible viscous flow problems. *J. Comput. Phys.*, 135(2):118–125, 1997.

[30] A. J. Chorin and J. E. Marsden. *A mathematical introduction to fluid mechnanics: third edition*. Spinger-Verlag: New York, 1993.

[31] P. Colella, D. T. Graves, B. J. Keen, and D. Modiano. A Cartesian grid embedded boundary method for hyperbolic conservation laws. *J. Comp. Phys.*, 211:347–366, January 2006.

[32] P. Colella, D. T. Graves, T. J. Ligocki, D. F. Martin, D. Modiano, D. B. Serafini, and B. Van Straalen. Chombo Software Package for AMR Applications - Design Document. unpublished, 2000.

[33] P. Colella and D. F. Martin. Dan's nifty paper on AMRINS. unpublished, 2004.

[34] P. Colella and P. R. Woodward. The piecewise parabolic method (PPM) for gas-dynamical simulations. *J. Comput. Phys.*, 54:174–201, 1984.

[35] R.K. Crockett, P. Colella, R. Fisher, R.I. Klein, and C.F. McKee. An unsplit, cell-centered godunov method for ideal mhd. *J. Comput. Phys.*, 203:422–448, 2005.

[36] Patrick F. Cummins, Laurence Armi, and Svein Vagle. Upstream Internal Hydraulic Jumps. *Journal of Physical Oceanography*, 2006. In prep.

[37] Sharen J. Cummins and Murray Rudman. An sph projection method. *J. Comput. Phys.*, 152(2):584–607, 1999.

[38] T. Dauxios, A. Didier, and E. Falcon. Observation of near-critical reflection of internal waves in a stably stratified fluid. *Phys. Fluids*, 16:1936–41, 2004.

[39] J. K. Dukowicz and R. D. Smith. Implicit free-surface method for the bryan-cox-semtner ocean model. *J. Geophys. Res.*, 99:7991–8014, 1994.

[40] C. C. Eriksen. Implications of ocean bottom reflection for internal waves spectra and mixing. *J. Phys. Oceanogr.*, 15:1145–156, 1985.

[41] David Farmer and Laurence Armi. The Generation and Trapping of Solitary Waves over Topography. *Science*, 283(5399):188–190, 1999.

[42] A. D. Fox and S.J. Maskell. Two-way interactive nesting of primitive equation ocean models with topography. *J. Phys. Oceanogr.*, 25:2977–2996, 1995.

[43] O. B. Fringer and D. D. Holm. Integrable vs. nonintegrable geodesic soliton behavior. *Physica D*, 150:237–263, 2001.

[44] O. B. Fringer and R. L. Street. The dynamics of breaking progressive interfacial waves. *J. Fluid Mech.*, 494:319–353, 2003.

[45] N. K. Ganju, D. H. Schoellhamer, J. C. Warner, M. F. Barad, and S. G. Schladow. Tidal oscillation of sediment between a river and a bay: a conceptual model. *Estuarine Coastal and Shelf Science*, 60:81–90, May 2004.

[46] C. Gatti-Bono and P. Colella. An anelastic allspeed projection method for gravitationally stratified flows. *J. Comp. Phys.*, 216(2):589–615, August 2006.

[47] Frederic Gibou, Ronald P. Fedkiw, Li-Tien Cheng, and Myungjoo Kang. A second-order-accurate symmetric discretization of the Poisson equation on irregular domains. *J. Comput. Phys.*, 176(1):205–227, 2002.

[48] D. T. Graves. *An Approximate Projection Method Suitable for the Modeling of Rapidly Rotating Flows.* PhD thesis, Dept. of Mechanical Engineering, Univ. of California, Berkeley, December 1996.

[49] Richard J. Greatbatch, Youyu Lu, and Yi Cai. Relaxing the Boussinesq Approximation in Ocean Circulation Models. *Journal of Atmospheric and Oceanic Technology*, 18(11), November 2001.

[50] P. M. Gresho, S. T. Chan, R. L. Lee, and C. D. Upson. A modified finite element method for solving the time-dependent, incompressible Navier-Stokes equations. I Theory. *International Journal for Numerical Methods in Fluids*, 4:557–598, June 1984.

[51] Boyce E. Griffith and Charles S. Peskin. On the order of accuracy of the immersed boundary method: higher order convergence rates for sufficiently smooth problems. *J. Comput. Phys.*, 208(1):75–105, 2005.

[52] K. R. Helfrich. Internal solitary wave breaking and run up on a uniform slope. *J. Fluid Mech.*, 243:133–154, 1992.

[53] T. Hibiya. Generation mechanism of internal waves by tidal flow over a sill. *J. Geophys. Res.*, 91(C6):7697–7708, 1986.

[54] T. Hibiya. The generation of internal waves by tidal flow over Stellwagen Bank. *J. Geophys. Res.*, 93(C1):533–542, 1988.

[55] C. W. Hirt, J. L. Cook, and T. D. Butler, editors. *A Lagrangian method for calculating the dynamics of an incompressible fluid with free surface*, 1972.

[56] P. Hosegood, J. Bonnin, and H. van Haren. Solibore-induced sediment resuspension in the faeroe-shetland channel. *Geophys. Res. Lett.*, 31(9):L09301, 2004.

[57] G. N. Ivey and R. I. Nokes. Vertical mixing due to the breaking of critical internal waves on sloping boundaries. *J. Fluid Mech.*, 204:479–500, 1989.

[58] C. Jackson. An Atlas of Internal Solitary-like Waves and their Properties, 2004. www.internalwaveatlas.com, 2nd edition.

[59] H. Johansen and P. Colella. A Cartesian grid embedded boundary method for Poisson's equation on irregular domains. *J. Comput. Phys.*, 147(2):60–85, December 1998.

[60] Hans Johansen and Phillip Colella. A Cartesian grid embedded boundary method for Poisson's eqaution on irregular domains. *J. Comput. Phys.*, 1998.

[61] Hans Svend Johansen. *Cartesian Grid embedded Boundary Finite Difference Methods for Elliptic and Parabolic Partial Differential Equations on Irregular Domains.* PhD thesis, University of California, Berkeley, 1997.

[62] Peter D. Killworth, David Stainforth, David J. Webb, and Stephen M. Paterson. The Development of a Free-Surface Bryan-Cox-Semtner Ocean Model. *Journal of Physical Oceanography*, 21(9):1333–1348, 1991.

[63] J. M. Klymak and J. N. Moum. Internal solitary waves of elevation advancing on a shoaling shelf. *Geophys. Res. Lett.*, 30(20):2045, 2003.

[64] M. F. Lai. *A Projection Method for Reacting Flow in the Zero Mach Number Limit.* PhD thesis, University of California, Berkeley, 1994.

[65] K. G. Lamb. Numerical experiments of internal wave generation by strong tidal flow across a finite amplitude bank edge. *J. Geophys. Res.*, 99(C1):843–864, 1994.

[66] K. G. Lamb and B. Wan. Conjugate flows and flat solitary waves for a continuously stratified fluid. *Physics of Fluids*, 10:2061–2079, August 1998.

[67] C. Lee and R. Beardsley. The generation of lone nonlinear internal waves in a weakly stratified shear flow. *J. Geophys. Res.*, 79:453–462, 1974.

[68] M. S. Longuet-Higgins and E. D. Cokelet. The deformation of steep surface waves on water. i. a numerical method of computation. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 350(1660), July 1976.

[69] R. Lowe, P. Linden, and J. Rottman. A laboratory study of the velocity structure in an intrusive gravity current. *Journal of Fluid Mechanics*, 456:33–48, April 2002.

[70] J. Marshall, C. Hill, L. Perelman, and A. Adcroft. Hydrostatic, quasi-hydrostatic, and nonhydrostatic ocean modeling. *J. Geophys. Res.*, 102:5733–5752, 1997.

[71] D. F. Martin and K. L. Cartwright. Solving Poisson's equation using adaptive mesh refinement. *Technical Report UCB/ERI M96/66 UC Berkeley*, 1996.

[72] Daniel Francis Martin. *An Adaptive Cell-Centered Projection Method for the Incompressible Euler Equations*. PhD thesis, Dept. of Mechanical Engineering, Univ. of California, Berkeley, 1998.

[73] T. Matsuura and T. Hibiya. An experimental and numerical study of the internla wave generation by tide-topography interaction. *J. Phys. Oceanogr.*, 20:506–521, 1990.

[74] T. Maxworthy. A note on internal solitary waves produced by tidal flow over a three dimensional ridge. *J. Geophys. Res.*, 84:338–346, 1979.

[75] P. McCorquodale, P. Colella, and H. Johansen. A Cartesian grid embedded boundary

method for the heat equation on irregular domains. *J. Comput. Phys.*, 173:620–635, November 2001.

[76] Peter McCorquodale, Phillip Colella, Gregory T. Balls, and Scott B. Baden. A scalable parallel Poisson solver in three dimensions with infinite-domain boundary conditions. In *Proceedings of the 7th International Workshop on High Performance Scientific and Engineering Computing (HPSEC-05)*, pages 814–822, Oslo, Norway, June 2005.

[77] H. Michallet and G. N. Ivey. Experiments on mixing due to internal solitary waves breaking on uniform slopes. *J. Geophys. Res.*, 104:13,467–13,477, 1999.

[78] Michael Minion. *Two Methods for the Study of Vortex Patch Evolution on Locally Refined Grids.* PhD thesis, U.C. Berkeley, May 1994.

[79] J. J. Monaghan. Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics*, 30:543–574, 1992.

[80] J. N. Moum, D. M. Farmer, W. D. Smyth, L. Armi, and S. Vagle. Structure and generation of turbulence at interfaces strained by internal solitary waves propagating shoreward over the continental shelf. *J. Phys. Oceanogr.*, 33:2093–2112, 2003.

[81] W. Munk. Abyssal recipes. *Deep-Sea Res.*, 13:707–730, 1966.

[82] W. Munk and C. Wunsch. Abyssal recipes II: energetics of tidal and wind mixing. *Deep-Sea Res.*, 45:1977–2010, 1998.

[83] R. B. Pember, J. B. Bell, P. Colella, and W. Y. Crutchfield. An adaptive Cartesian grid

method for unsteady compressible flow in irregular regions. *Journal of Computational Physics*, 120(2):278–304, September 1995.

[84] R.B. Pember, L.H. Howell, J.B. Bell, P. Colella, W.Y. Crutchfield, W.A. Fiveland, and J.P. Jessee. An adaptive projection method for unsteady, low-mach number combustion. *Comb. Sci. Tech.*, 140:123–168, 1998.

[85] P. Penven, L. Debreu, P. Marchesiello, and J.C. McWilliams. Evaluation and application of the roms 1-way embedding procedure to the central california upwelling system. *Ocean Modelling*, 2005. in press.

[86] Charles S. Peskin. The immersed boundary method. *Acta Numerica*, 11:479–517, 2002.

[87] E. G. Puckett, A. S. Almgren, J. B. Bell, D. L. Marcus, and W. J. Rider. A high-order projection method for tracking fluid interfaces in variable density incompressible flows. *J. Comput. Phys.*, 130:269–282, 1997.

[88] R. D. Ray and G. T. Mitchum. Surface manifestation of internal tides generated near hawaii. *Geophys. Res. Lett.*, 23:2101–2116, 1996.

[89] W. J. Rider. Filtering non-solenoidal modes in numerical solutions of incompressible flows. *International Journal for Numerical Methods in Fluids*, 28(5):789–814, 1998.

[90] Francisco J. Rueda. *A Three-Dimensional Hydrodynamic and Transport Model for Lake Environments*. PhD thesis, University of California, Davis, 2001.

[91] S. G. Schladow, S. Ó. Pálmarsson, T. E. Steissberg, S. J. Hook, and F. E. Prata.

An extraordinary upwelling event in a deep thermally stratified lake. *Geophysical Research Letters*, 31:15504–+, 2004.

[92] P. O. Schwartz, M. F. Barad, P. Colella, and T. J. Ligocki. A Cartesian grid embedded boundary method for the heat equation and Poisson's equation in three dimensions. *J. Comp. Phys.*, 211(2):531–550, 2006.

[93] A. Scotti and J. Pineda. Observations of very large and steep internal waves of elevation near the Massachussetts coast. *Geophys. Res. Lett.*, 31, 2004. L22307.

[94] W.C. Skamarock and J.B. Klemp. Adaptive grid refinement for two-dimensional and three-dimensional nonhydrostatic atmospheric flow. *Monthly Weather Review*, 121:788–804, 1993.

[95] Peter E. Smith. *A Three-Dimensional, finite-difference model for estuarine circulation.* PhD thesis, University of California, Davis, 1997.

[96] T. P. Stanton and L. A. Ostrovsky. Observations of highly nonlinear interal solitons over the continental shelf. *Geophys. Res. Lett.*, 25:2695–2698, 1998.

[97] T. E. Steissberg, S. J. Hook, and S. G. Schladow. Characterizing partial upwellings and surface circulation at Lake Tahoe, California-Nevada, USA with thermal infrared images. *Remote Sensing of Environment*, 99:2–15, 2005.

[98] T. E. Steissberg, S. J. Hook, and S. G. Schladow. Measuring surface currents in lakes with high spatial resolution thermal infrared imagery. *Geophysical Research Letters*, 32:11402–+, 2005.

[99] M. Sussman and E. G. Puckett. A coupled level set and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows. *J. Comput. Phys.*, 162(2):301–337, 2000.

[100] M. M. Sussman, A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. Welcome. An adaptive level set approach for incompressible two-phase flows. *J. Comp. Phys.*, 148:81–124, 1999.

[101] S. A. Thorpe. Recent developments in the study of ocean turbulence. *Annu. Rev. Earth Planet. Sci..*, 32:91–109, 2004.

[102] David Paul Trebotich. *A Projection Method for Incompressible Viscous Flow on a Deformable Domain*. PhD thesis, Dept. of Mechanical Engineering, Univ. of California, Berkeley, December 1998.

[103] C. D. Troy and J. R. Koseff. The instability and breaking of long internal waves. *J. Fluid Mech.*, 2005. in press.

[104] E.H. Twizell, A.B. Gumel, and M.A. Arigu. Second-order, l_0-stable methods for the heat equation with time-dependent boundary conditions. *Advances in Computational Mathmatics*, 6:333–352, 1996.

[105] S. K. Venayagamoorthy and O. B. Fringer. Nonhydrostatic and nonlinear contributions to the energy flux budget in nonlinear internal waves. *Geophys. Res. Lett.*, 32, 2005. L15603.

[106] D.-L. Zhang, H.-R. Chang, N. L. Seaman, T. T. Warner, and J. M. Fritsch. A two-way

interactive nesting procedure with variable domain resolution. *Mon. Weather Rev.*, 114:1330–1339, 1986.

Appendix A

Miscellaneous Discretizations

A.1 Matrix Form of the Laplacian Without EB's

The second-order accurate two-dimensional, anisotropic grid, Laplacian stencil can be written as:

$$L\phi = \begin{bmatrix} & \frac{1}{dy^2} & \\ \frac{1}{dx^2} & (\frac{-2}{dx^2} + \frac{-2}{dy^2}) & \frac{1}{dx^2} \\ & \frac{1}{dy^2} & \end{bmatrix} \phi \tag{A.1}$$

Where the stencil is simplified to:

$$L\phi = \begin{bmatrix} & b & \\ a & d & a \\ & b & \end{bmatrix} \phi \tag{A.2}$$

Which for a four by four periodic domain yields the following symmetric matrix operator:

$$
L\phi = \begin{bmatrix}
d & a & & a & b & & & & & & & b & & & \\
a & d & a & & & b & & & & & & & b & & \\
 & a & d & a & & & b & & & & & & & b & \\
a & & a & d & & & & b & & & & & & & b \\
b & & & & d & a & & a & b & & & & & & \\
 & b & & & a & d & a & & & b & & & & & \\
 & & b & & & a & d & a & & & b & & & & \\
 & & & b & a & & a & d & & & & b & & & \\
 & & & & b & & & & d & a & & a & b & & \\
 & & & & & b & & & a & d & a & & & b & \\
 & & & & & & b & & & a & d & a & & & b \\
 & & & & & & & b & a & & a & d & & & \\
b & & & & & & & & b & & & & d & a & & a \\
 & b & & & & & & & & b & & & a & d & a & \\
 & & b & & & & & & & & b & & & a & d & a \\
 & & & b & & & & & & & & b & a & & a & d
\end{bmatrix}
\begin{bmatrix}
\phi_{0,0} \\ \phi_{1,0} \\ \phi_{2,0} \\ \phi_{3,0} \\ \phi_{0,1} \\ \phi_{1,1} \\ \phi_{2,1} \\ \phi_{3,1} \\ \phi_{0,2} \\ \phi_{1,2} \\ \phi_{2,2} \\ \phi_{3,2} \\ \phi_{0,3} \\ \phi_{1,3} \\ \phi_{2,3} \\ \phi_{3,3}
\end{bmatrix}
\tag{A.3}
$$

A.2 Line Solves for Elliptic and Parabolic Equations

One of the target application areas for this research is studying large-scale geophysical flows. These flows are typically of large aspect ratio, i.e. horizontal scales are larger than vertical scales. For example, Lake Tahoe has horizontal scales that are $O(10km)$

and a vertical scale that is $O(500m)$. When solving elliptic partial differential equations for large aspect ratio systems it is common to reduce the computational load by having $\Delta x \neq \Delta z \neq \Delta y$. When the aspect ratio is more than (roughly) 2, traditional multigrid smoothers fail to dampen high frequency errors and multigrid convergence stalls. This is a well known problem, and can be easily seen by noting that the stencil in (A.1) becomes dimensionally decoupled when $\Delta x >> \Delta y$. There are two "fixes" that are traditionally employed: 1) anisotropic coarsening in multigrid, 2) line solving, or 3) both [19]. For our approach in this research we chose to implement a line solver that is both AMR and EB aware. The AMR multigrid solution algorithm is described in Section 5.4. A pseudo-code

---

procedure LineSmoother(ϕ, R)

{

$\delta = 0$

$g = (I - \lambda L^h_{xy})\delta + \lambda R$

$\delta = (I + \lambda L^h_z)^{-1} g$

$\phi = \phi + \delta$

}

Figure A.1: Pseudo-code description of the line smoother algorithm.

description of our line smoother algorithm is given in Figure A.1. For the algorithm we initialize the correction δ to zero, then we evaluate our horizontal Laplacian stencil (L^h_{xy}),

and compute a new right-hand-side (g). Where, $\lambda = \frac{1}{4diag(L^h)}$, and where $(I - \lambda L^h_{xy})0 = 0$. Subsequently we do a tri-diagonal solve using our new right-hand-side ($g = \lambda R$). Then we apply the correction to ϕ (where ϕ is a correction since we are already in residual-correction form). We use the Thomas Algorithm when doing the tri-diagonal solve, and where we have covered cells ($\kappa = 0$) we set the diagonal term to 1 and the two off diagonal terms to zero. This decouples covered cells from the non-covered control volumes and permits the line solver to handle arbitrarily complex geometry. A validation of the line solver is given in Section B.2.

A.3 Stair-Step Errors

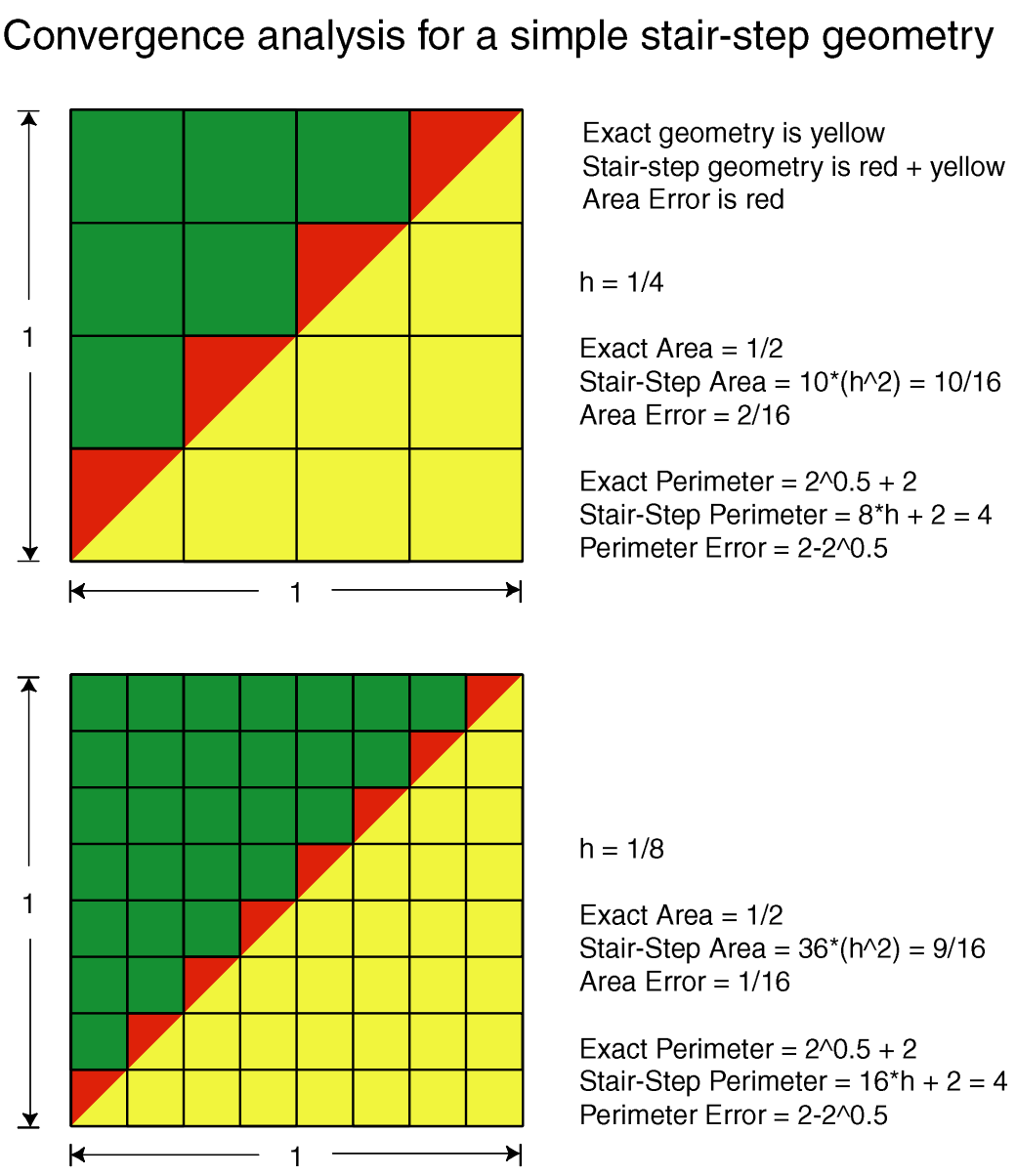

Figure A.2: Stair-Step accuracy

A.4 Free-Surface Discretization: Scalars

For a conservative treatment of scalars at the free-surface boundary, our algorithm keeps a budget of scalar transport into and out of the free-surface volume. To do this in a stable way, we use the redistribution ideas from Section (4.3). The algorithm is developed as follows. We temporarily define $\boldsymbol{i}$ as an index for the first control volume below the top domain face ($z = 0$). A conservative update for scalars (denoted ρ) in the free-surface at time $n + 1$ is,

$$\rho^C_{\boldsymbol{i}+\boldsymbol{e}_D} = \rho^n_{\boldsymbol{i}+\boldsymbol{e}_D} - \Delta t D^C_{\boldsymbol{i}+\boldsymbol{e}_D}. \tag{A.4}$$

Where $D^C_{\boldsymbol{i}+\boldsymbol{e}_D}$ is a conservative (and free-stream preserving) divergence for the free-surface control volume ($\boldsymbol{i} + \boldsymbol{e}_D$),

$$D^C_{\boldsymbol{i}+\boldsymbol{e}_D} = \frac{w_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_D}\rho_{\boldsymbol{i}+\boldsymbol{e}_D} - w_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_D}\rho_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_D}}{\eta}, \tag{A.5}$$

and

$$\rho_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_D} = \begin{cases} \tilde{\rho}^L_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_D} & \text{if } w_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_D} > 0, \\ \rho_{\boldsymbol{i}+\boldsymbol{e}_D} & \text{if } w_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_D} < 0, \\ \frac{1}{2}(\tilde{\rho}^L_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_D} + \rho_{\boldsymbol{i}+\boldsymbol{e}_D}) & \text{if } w_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_D} = 0. \end{cases} \tag{A.6}$$

Since η can be zero, we need a stable update for scalars,

$$\rho^{NC}_{\boldsymbol{i}+\boldsymbol{e}_D} = \rho^n_{\boldsymbol{i}+\boldsymbol{e}_D} - \Delta t \left[\beta D^C_{\boldsymbol{i}+\boldsymbol{e}_D} + (1-\beta) D^{NC}_{\boldsymbol{i}+\boldsymbol{e}_D} \right], \tag{A.7}$$

where η is at time $n + \frac{1}{2}$ here and for the remainder of this Section. We use

$$D^{NC}_{\boldsymbol{i}+\boldsymbol{e}_D} = D_{\boldsymbol{i}}, \tag{A.8}$$

where $D_{\boldsymbol{i}}$ is our "standard" advective term from Section (4.3). The difference between (A.4) and (A.7) is

$$\rho^{C}_{\boldsymbol{i}+\boldsymbol{e}_D} - \rho^{NC}_{\boldsymbol{i}+\boldsymbol{e}_D} = \Delta t\Big[(\beta-1)D^{C}_{\boldsymbol{i}+\boldsymbol{e}_D} + (1-\beta)D^{NC}_{\boldsymbol{i}+\boldsymbol{e}_D}\Big]. \tag{A.9}$$

Where this is a mass if we multiply by the free-surface volume $|V_{\boldsymbol{i}+\boldsymbol{e}_D}| = \eta\Delta x\Delta y\alpha$ (where α is the area fraction at the domain face) and re-arrange

$$\delta M = \Delta t|V_{\boldsymbol{i}+\boldsymbol{e}_D}|\Big[(1-\beta)(D^{NC}_{\boldsymbol{i}+\boldsymbol{e}_D} - D^{C}_{\boldsymbol{i}+\boldsymbol{e}_D})\Big]. \tag{A.10}$$

We need to redistribute the mass (δM), and so we do this in a volume weighted manner,

$$\delta M_{\boldsymbol{i}+\boldsymbol{e}_D} = \frac{|V_{\boldsymbol{i}+\boldsymbol{e}_D}|}{|V_{\boldsymbol{i}}| + |V_{\boldsymbol{i}+\boldsymbol{e}_D}|}\delta M \tag{A.11}$$

and

$$\delta M_{\boldsymbol{i}} = \frac{|V_{\boldsymbol{i}}|}{|V_{\boldsymbol{i}}| + |V_{\boldsymbol{i}+\boldsymbol{e}_D}|}\delta M \tag{A.12}$$

This yields our conservative free-surface scalar update

$$\rho^{n+1}_{\boldsymbol{i}+\boldsymbol{e}_D} = \rho^{NC}_{\boldsymbol{i}+\boldsymbol{e}_D} + \frac{\delta M_{\boldsymbol{i}+\boldsymbol{e}_D}}{|V_{\boldsymbol{i}+\boldsymbol{e}_D}|}, \tag{A.13}$$

or

$$\rho^{n+1}_{\boldsymbol{i}+\boldsymbol{e}_D} = \rho^{NC}_{\boldsymbol{i}+\boldsymbol{e}_D} + \Delta t\frac{\alpha\eta}{\kappa\Delta z + \alpha\eta}\Big[(1-\beta)(D^{NC}_{\boldsymbol{i}+\boldsymbol{e}_D} - D^{C}_{\boldsymbol{i}+\boldsymbol{e}_D})\Big]. \tag{A.14}$$

For conservation, we need to modify the scalar inside of the domain face by,

$$\rho^{n+1}_{\boldsymbol{i}} = \rho^{C}_{\boldsymbol{i}} + \frac{\delta M_{\boldsymbol{i}}}{|V_{\boldsymbol{i}}|}, \tag{A.15}$$

or

$$\rho^{n+1}_{\boldsymbol{i}} = \rho^{C}_{\boldsymbol{i}} + \Delta t\frac{\alpha\eta}{\kappa\Delta z + \alpha\eta}\Big[(1-\beta)(D^{NC}_{\boldsymbol{i}+\boldsymbol{e}_D} - D^{C}_{\boldsymbol{i}+\boldsymbol{e}_D})\Big], \tag{A.16}$$

where as in Section (4.3),

$$\rho_{\boldsymbol{i}}^{C} = \rho_{\boldsymbol{i}}^{n} - \Delta t D_{\boldsymbol{i}}. \tag{A.17}$$

It is important to note in (A.14) and (A.16), D^C is multiplied by η, eliminating the division by η, a possibly zero value. If we examine (A.14) and (A.16) we notice that when $\alpha\eta = -\kappa\Delta z$ we divide by zero (because the volume weighted redistribution has a net receiving volume of zero). To fix this problem we define β as follows:

$$\beta = \begin{cases} 1 & \text{if } |\alpha\eta| > \frac{\kappa\Delta z}{2}, \\ \frac{2|\alpha\eta|}{\kappa\Delta z} & \text{else.} \end{cases} \tag{A.18}$$

This choice of β restricts our CFL to smaller than $\frac{1}{2}$.

A.5 Free-Surface Discretization: η and $< \vec{u} >$

Here we evolve (2.10) and (2.14) explicitly using stable time step(s) from time t^n to time t^{n+1}. Note that a "stable time step" might need to be smaller than $t^{n+1} - t^n$, i.e. we subcycle the free-surface boundary condition as needed (similar to [62, 39]). Subcycling permits us to evolve the momentum balance independent of the free-surface wave speed. Without subcycling, the C.F.L. constraint imposed by fast moving free-surface waves typically slows the main code down by ~ 100. Notice that (2.10) and (2.14) make up a coupled hyperbolic system. We advance this system at a subcycled time step $\Delta s = \Delta t/N$, where the integer $N >= 1$, and ensures that our CFL constraint on free-surface wave speeds is satisfied. We advance (2.10) and (2.14) by

$$\eta^{m+1} = \eta^{m} - \Delta s \nabla^{\perp} \cdot ((\eta + H) < \vec{u} >) \tag{A.19}$$

and

$$< \vec{u}^{\perp} >^{m+1} = \vec{u}^{m} + \Delta s \nabla^{\perp}(g\eta) \tag{A.20}$$

Where $\nabla^{\perp} \cdot ((\eta + H) < \vec{u} >)$ and $\nabla^{\perp}(g\eta)$ are computed with a first-order Godunov method, where g is the gravitational acceleration. In the Godunov method, for the $d = 0$ component, given low (L) and high (H) states, our Riemann solver is as follows,

$$< u >^{*} = \frac{1}{2}\left[< u >_L + < u >_H + \sqrt{\frac{-g}{h}}(\eta_L - \eta_H) \right] \tag{A.21}$$

and

$$\eta^{*} = \frac{1}{2}\left[\eta_L + \eta_H + \sqrt{\frac{h}{-g}}(< u >_L - < u >_H) \right]. \tag{A.22}$$

Appendix B

Convergence Properties of the Main Operators

B.1 Hyperbolic Equations

Our test problem here is scalar advection ($s_t + \vec{u} \cdot \nabla s = 0$) of a Gaussian bump down an inclined channel. A 2D isotropic convergence study for this problem is shown in Table B.1. A 2D AMR isotropic convergence study for this problem is shown in Table B.2. A 2D anisotropic convergence study for this problem is shown in Table B.3. A 3D anisotropic convergence study for this problem is shown in Table B.4.

For hyperbolic problems we expect the convergence rates for our method to be: $r_\infty = 1.0, r_2 = 1.5, r_1 = 2.0$, where we indicate convergence rates r_p as in (6.2). The convergence tables in this section indicate that we are achieving the expected rates.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
L_∞ Norm of Scalar Error	8.3294e-03	4.1744e-03	**9.9664e-01**
L_1 Norm of Scalar Error	2.2876e-05	5.6418e-06	**2.0196e+00**
L_2 Norm of Scalar Error	6.6815e-05	1.9502e-05	**1.7765e+00**

Table B.1: Hyperbolic test problem: isotropic grid, solution error and convergence rates: $\Delta x_c = \frac{1}{256} = 2\Delta x_m = 4\Delta x_f$; a $2D$ calculation.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
L_∞ Norm of Scalar Error	1.3367e-02	5.7748e-03	**1.2109e+00**
L_1 Norm of Scalar Error	1.5316e-04	3.5190e-05	**2.1218e+00**
L_2 Norm of Scalar Error	5.7150e-04	1.3468e-04	**2.0852e+00**

Table B.2: Hyperbolic test problem: isotropic grid, solution error and convergence rates: $\Delta x_c = \frac{1}{16} = 2\Delta x_m = 4\Delta x_f$; a $2D$ calculation with 3 AMR levels having nref = 4.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
L_∞ Norm of Scalar Error	1.6868e-02	8.4677e-03	**9.9426e-01**
L_1 Norm of Scalar Error	1.1499e-04	3.0643e-05	**1.9078e+00**
L_2 Norm of Scalar Error	5.7566e-04	2.0620e-04	**1.4812e+00**

Table B.3: Hyperbolic test problem: anisotropic grid ($\Delta x = 4\Delta y$), solution error and convergence rates: $\Delta x_c = \frac{1}{128} = 2\Delta x_m = 4\Delta x_f$; a $2D$ calculation.

Variable	Medium-Coarse Error	Fine-Medium Error	Order
L_∞ Norm of Scalar Error	5.4763e-02	3.2235e-02	**7.6459e-01**
L_1 Norm of Scalar Error	8.4448e-03	2.1650e-03	**1.9637e+00**
L_2 Norm of Scalar Error	1.7622e-02	4.4385e-03	**1.9893e+00**

Table B.4: Hyperbolic test problem: anisotropic grid ($\Delta x = \Delta y = 2\Delta z$), solution error and convergence rates: $\Delta x_c = \frac{1}{16} = 2\Delta x_m = 4\Delta x_f$; a $3D$ calculation. Computed using 2 parallel processors.

B.2 Elliptic Equations

To illustrate our ability to solve anisotropic elliptic problems, here we present results from a simple test case, where the exact solution is $\phi = sin(1.3x)sin(2.2y)[sin(3.1z)]$. Consider a sphere of radius 0.25 centered in a unit domain, see the coarse-grid layout in figure B.1. Our 2D fine grid has 32x640 (i.e. 20:1 aspect ratio) level-zero cells (including covered cells) and our AMR refinement ratio is two. The coarse solution is on the same grid layout as the fine solution, but coarsened by a factor of two.

We solve an elliptic problem using our AMR multigrid method (outlined above), and present the expected multigrid convergence history in figure B.2. The method maintains second-order accuracy, see Table B.5 for 2D AMR results, and see Table B.6 for 3D single grid results.

For elliptic problems we expect the convergence rates for our method to be: $r_\infty = 2.0, r_2 = 2.0, r_1 = 2.0$, where we indicate convergence rates r_p as in (6.2). The convergence tables in this section indicate that we are achieving the expected rates.

B.3 Parabolic Equations

Our test problem here is scalar diffusion ($s_t = \nu \Delta s + H$) in a unit domain with a centered embedded sphere of radius $\frac{1}{4}$. Our exact solution $s = sin(5x)sin(5y)sin(5z)cos(t)$ is imposed as an initial condition (at $t = 0$) and as a Dirichlet boundary condition. The source term H is computed from the exact solution given $\nu = 0.01$. A 2D anisotropic convergence study for this problem is shown in Table B.7. A 3D anisotropic convergence study for this problem is shown in Table B.8.

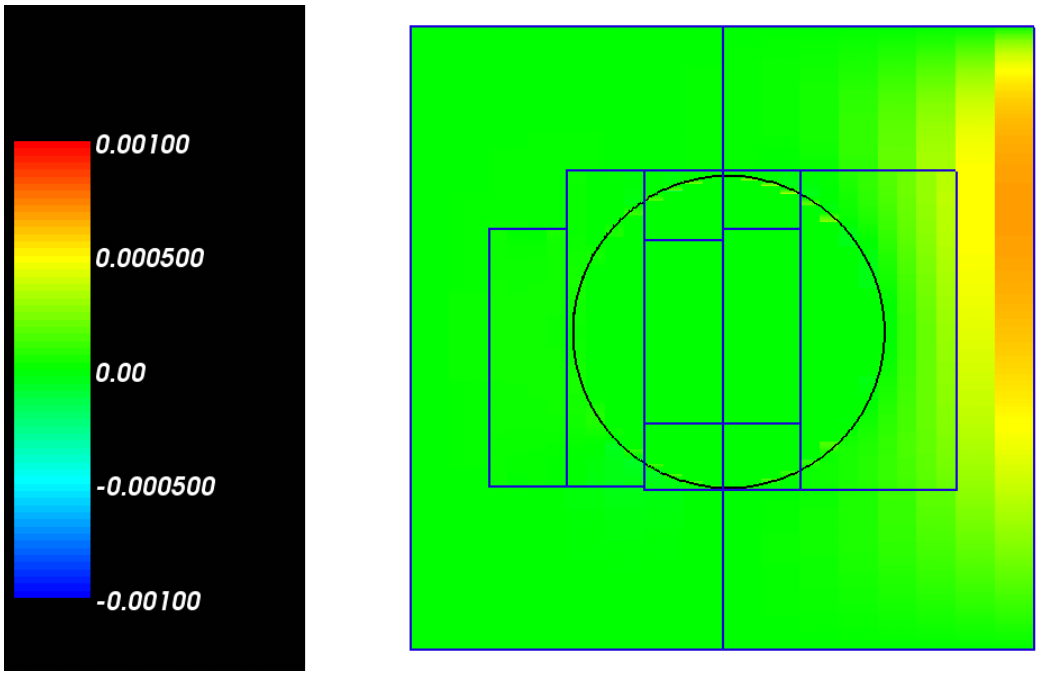

Figure B.1: Coarse solution error for a 20:1 aspect ratio 2D AMR solve

Variable	Coarse Error	Fine Error	Order
L_∞ Norm Error	1.051167e-03	2.836212e-04	**1.889955e+00**
L_1 Norm Error	9.717318e-05	2.456552e-05	**1.983924e+00**
L_2 Norm Error	1.909830e-04	4.851628e-05	**1.976903e+00**

Table B.5: Elliptic test problem: solution error convergence rates for a 20:1 aspect ratio AMR solve. $\Delta x_f = \frac{1}{32}$ and $\Delta x_c = 2\Delta x_f$, $2D$

Variable	Coarse Error	Fine Error	Order
L_∞ Norm Error	3.873316e-03	1.019814e-03	**1.925263e+00**
L_1 Norm Error	6.459656e-05	1.403111e-05	**2.202828e+00**
L_2 Norm Error	9.121400e-05	2.166892e-05	**2.073628e+00**

Table B.6: Elliptic test problem: solution error convergence rates for a 4:4:1 aspect ratio single grid solve. $\Delta x_f = \frac{1}{64}$ and $\Delta x_c = 2\Delta x_f$, $3D$

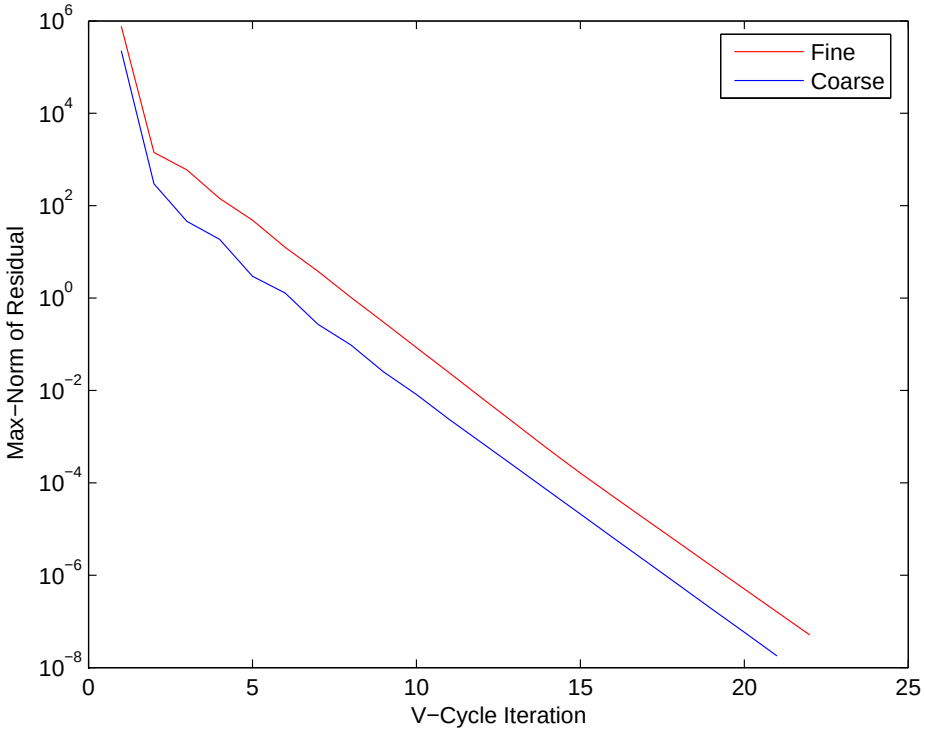

Figure B.2: Multigrid convergence for a 20:1 aspect ratio 2D AMR solve

For parabolic problems we expect the convergence rates for our method to be: $r_\infty = 2.0, r_2 = 2.0, r_1 = 2.0$, where we indicate convergence rates r_p as in (6.2). The convergence tables in this section indicate that we are achieving the expected rates.

B.4 Density Weighted Projections

Here we project a given velocity field onto a divergence-free space. Our given divergent velocity field is $\vec{u} = (sin(\pi x), sin(2\pi y), sin(3\pi z))$. For the density weighted projections, which use DWPM1 of 4.1.4, we prescribe a density field of $\rho = 2 + sin(x) + sin(y) + sin(z)$. We have a unit domain with a centered embedded sphere of radius $\frac{1}{4}$, and our boundary conditions are $\vec{u} \cdot \vec{n} = 0$.

B.4.1 Density Weighted Cell Centered Projection

A 2D anisotropic convergence study for this problem is shown in Table B.9. A 3D anisotropic convergence study for this problem is shown in Table B.10. For approximate cell-centered projections we expect the convergence rates for our method to be: $r_\infty = 1.0, r_2 = 1.5, r_1 = 2.0$, where we indicate convergence rates r_p as in (6.2). The convergence tables in this section indicate that we are achieving the expected rates.

B.4.2 Density Weighted MAC Projection

A 2D anisotropic convergence study for this problem is shown in Table B.11. A 3D anisotropic convergence study for this problem is shown in Table B.12. For MAC-projections we expect exact projections, and this is what we are achieving.

Variable	Coarse Error	Fine Error	Order
L_∞ Norm of Error	3.628538e-02	1.074547e-02	**1.755660e+00**
L_1 Norm of Error	5.242795e-03	1.335954e-03	**1.972465e+00**
L_2 Norm of Error	7.953677e-03	2.038763e-03	**1.963928e+00**

Table B.7: Parabolic test problem: solution error convergence rates ($\Delta x = 2\Delta y$). $\Delta x_f = \frac{1}{16}$ and $\Delta x_c = 2\Delta x_f$; a $2D$ calculation with 2 AMR levels having nref = 4.

Variable	Coarse Error	Fine Error	Order
L_∞ Norm of Error	4.814209e-02	1.298819e-02	**1.890098e+00**
L_1 Norm of Error	4.869321e-03	1.212166e-03	**2.006133e+00**
L_2 Norm of Error	7.659393e-03	1.922383e-03	**1.994334e+00**

Table B.8: Parabolic test problem: solution error convergence rates ($\Delta x = \Delta y = 2\Delta z$). $\Delta x_f = \frac{1}{16}$ and $\Delta x_c = 2\Delta x_f$; a $3D$ calculation with 1 AMR levels having nref = 4.

Variable	Coarse Divergence	Fine Divergence	Order
L_∞ Norm Divergence	1.050206e-01	5.342729e-02	**9.750238e-01**
L_1 Norm Divergence	3.527372e-03	9.168277e-04	**1.943871e+00**
L_2 Norm Divergence	6.020600e-03	2.022201e-03	**1.573981e+00**

Table B.9: Cell centered projection: divergence and convergence rates ($\Delta x = 2\Delta y$). $\Delta x_f = \frac{1}{128}$ and $\Delta x_c = 2\Delta x_f$; a $2D$ calculation with 2 AMR levels having nref = 2.

Variable	Coarse Divergence	Fine Divergence	Order
L_∞ Norm Divergence	1.299411e+00	6.439040e-01	**1.012940e+00**
L_1 Norm Divergence	1.519918e-01	4.303184e-02	**1.820517e+00**
L_2 Norm Divergence	2.056353e-01	6.222668e-02	**1.724483e+00**

Table B.10: Cell centered projection: divergence and convergence rates ($\Delta x = \Delta y = 2\Delta z$). $\Delta x_f = \frac{1}{32}$ and $\Delta x_c = 2\Delta x_f$; a $3D$ calculation with 2 AMR levels having nref = 2.

Variable	Coarse Divergence	Fine Divergence	Order
L_∞ Norm Divergence	1.054312e-11	1.187086e-10	**discretely exact**
L_1 Norm Divergence	2.742717e-13	2.923945e-12	**discretely exact**
L_2 Norm Divergence	9.290043e-13	8.457004e-12	**discretely exact**

Table B.11: MAC Projection: divergence and convergence rates ($\Delta x = 2\Delta y$). $\Delta x_f = \frac{1}{32}$ and $\Delta x_c = 2\Delta x_f$; a $2D$ calculation with 2 AMR levels having nref = 2.

Variable	Coarse Divergence	Fine Divergence	Order
L_∞ Norm Divergence	9.036771e-12	1.013891e-10	**discretely exact**
L_1 Norm Divergence	2.306771e-13	1.646531e-12	**discretely exact**
L_2 Norm Divergence	6.033329e-13	4.551875e-12	**discretely exact**

Table B.12: MAC Projection: divergence and convergence rates ($\Delta x = \Delta y = 2\Delta z$). $\Delta x_f = \frac{1}{32}$ and $\Delta x_c = 2\Delta x_f$; a $3D$ calculation with 2 AMR levels having nref = 2.

Appendix C

Examples

Here we present example visualizations to illustrate our ability to approximate complex geometries, and simulate interesting flows using the method of this work.

C.1 Grid Generation

Our first grid generation example is shown in Figure C.1 and is an approximation of a sphere-flake, a recursive flake like geometry. Again, it is important to note that in this and the following figures we visualize the EB's as piecewise planar, when in fact we use piecewise quadratic approximations for face intersections.

This example is generated from a digital elevation model of the South China Sea. In Figure C.2 we present South China Sea bathymetry with parallel adaptive mesh refinement boxes. Black boxes are a decomposition of the coarsest level, red boxes are finer grids. Each box is further sub-divided into individual control volumes. The red boxes refine both the Luzon Strait vicinity (right) and the Dongsha Island region (left). Upper right is Taiwan,

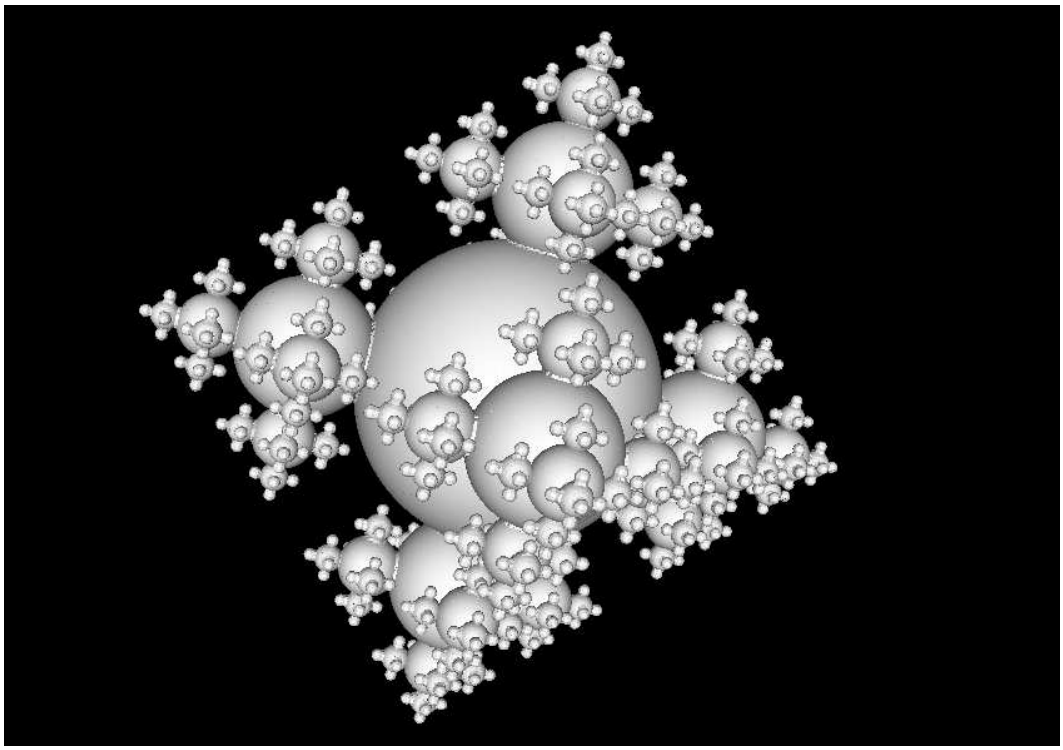

Figure C.1: EB approximation of a sphere-flake

lower right is the Philippines, mainland China is upper left. The East-West slice is colored by distance to the bed.

C.2 Flow Calculations

Here we present the classic lock-exchange problem from fluid dynamics. Flow is inside a 0.5 $[m]$ tall, by 3 $[m]$ wide tank. On the left side of the tank we start with light water, on the right is heavy water. The density ratio of light fluid to heavy fluid is 1000/1030. A resulting snapshot of the flow as it evolves using 4 AMR levels is shown (zoomed in) in Figure C.3, with the adaptive control volumes shown in Figure C.4. Notice that we

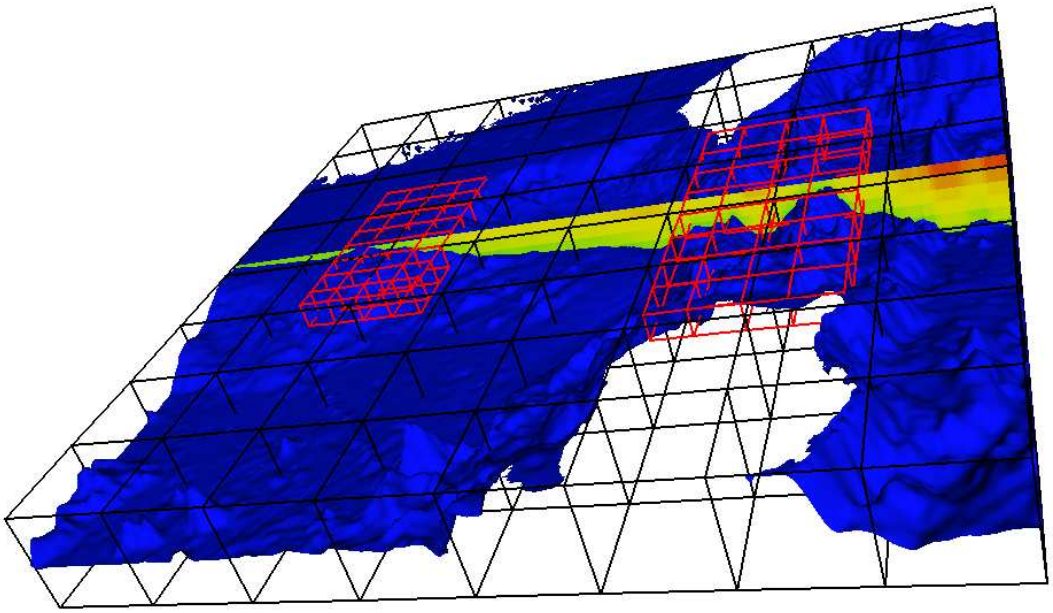

Figure C.2: EB approximation of the South China Sea

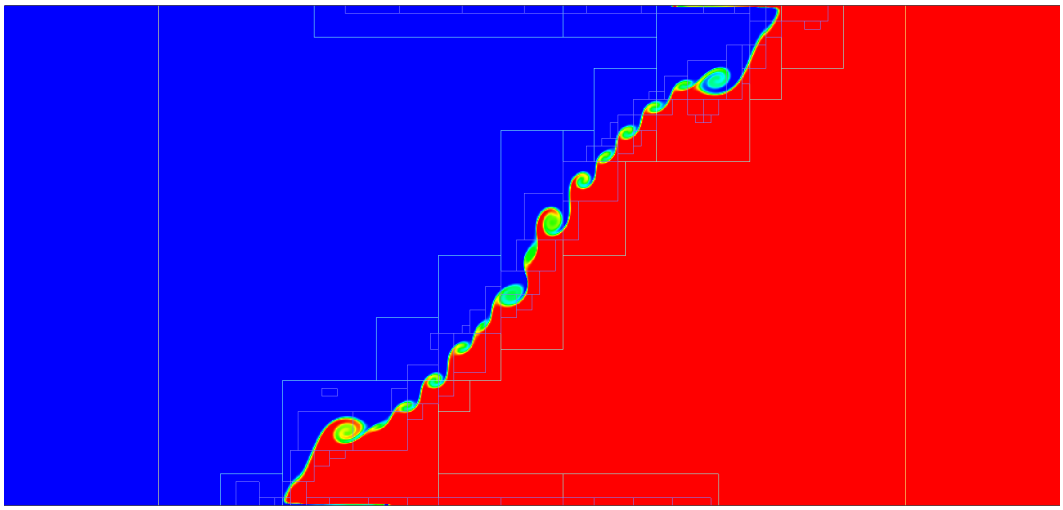

Figure C.3: Lock-exchange with AMR: density plot

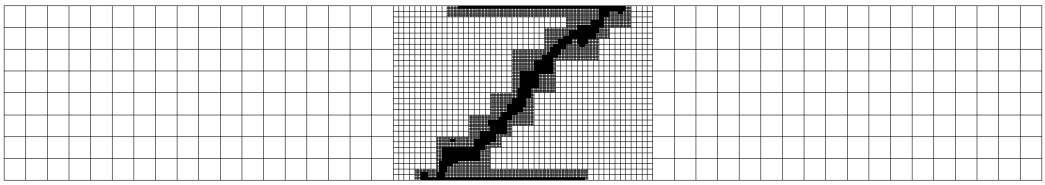

Figure C.4: Lock-exchange with AMR: control volumes

are resolving the boundary layer at the top and bottom of the tank, and are generating Kelvin-Helmholtz type billows along the unstable density interface. If we add a bottom slope to the same tank setup we see results as in figure C.5.

Next we present a simulation of an intrusive gravity current problem (from the lab study of [69]). The laboratory setup is a 1.82 $[m]$ long by 0.2 $[m]$ tall by 0.23 $[m]$ wide tank. The initially still water is separated into 3 density zones, where $\rho_1 = (\rho_0 + \rho_2)/2$. On the left 0.30 $[m]$ of the tank is ρ_1, the remaining right side of the tank is split vertically with

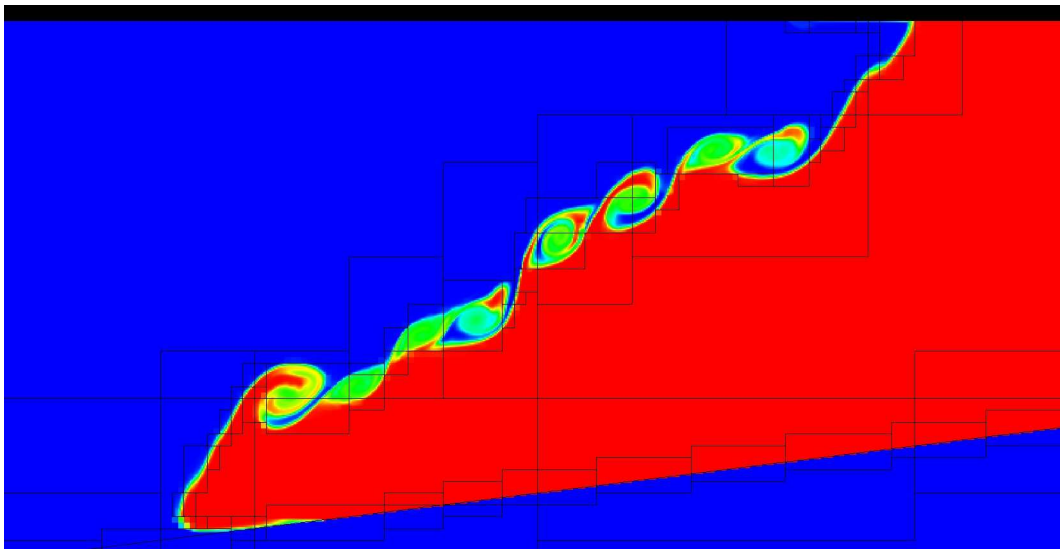

Figure C.5: Lock-exchange down a slope with AMR: density plot

ρ_0 on the top half and ρ_2 on the bottom half. We set $\rho_0 = 1002.6$ $[kg/m^3]$ and $\rho_2 = 1039.9$ $[kg/m^3]$ and let the flow evolve. Figure C.6 presents a snapshot of the 2D calculation.

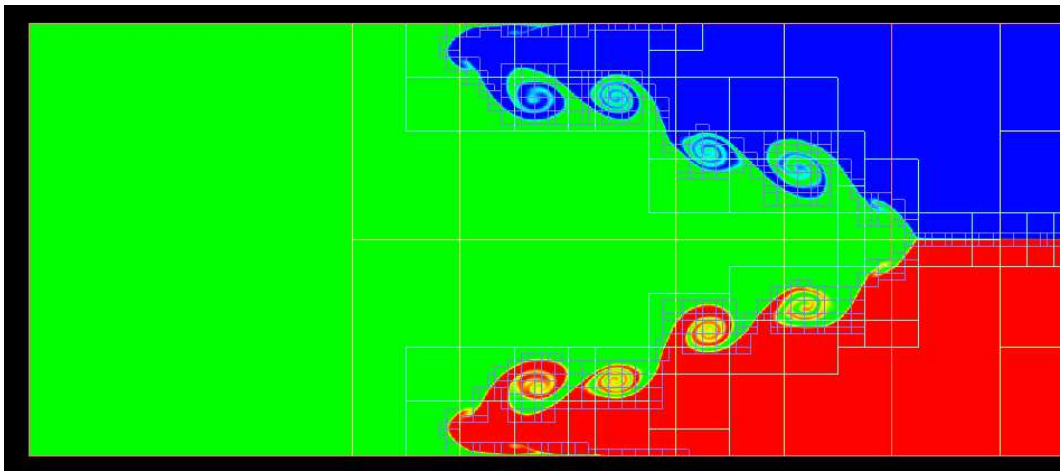

Figure C.6: Intrusive gravity current: density plot

Appendix D

Journal Papers

D.1 Tidal Oscillation of Sediment Between a River and a Bay: A Conceptual Model

During completion of this dissertation [45] was published.

D.2 A Fourth-Order Accurate Local Refinement Method for Poisson's Equation

During completion of this dissertation [5] was published.

D.3 A Cartesian Grid Embedded Boundary Method for the Heat Equation and Poisson's Equation in Three Dimensions

During completion of this dissertation [92] was published.

D.4 A Cartesian Grid Embedded Boundary Method for the Incompressible Navier-Stokes Equations

During completion of this dissertation [7] was in preparation for publication.

D.5 An Adaptive Cartesian Grid Embedded Boundary Method for the Incompressible Navier-Stokes Equations

During completion of this dissertation [6] was in preparation for publication.